Group Quiz 1 Report: Bookstore Database

Group Members:

- Rukundo Espoir [27678]
- Gatera K Jessica [27630]
- Shyaka Chriss [27889]

Instructor: : Maniraho Eric

Introduction

This report details the design and implementation of a relational database for a bookstore. The project was completed as part of a group assignment and demonstrates key database concepts, including table creation, the use of constraints, different types of joins, and the creation of indexes and views to optimize and simplify data access.

1. Database Design

We designed a relational database with three tables to manage a bookstore: Authors, Books, and Sales. These tables are related to each other to ensure data integrity and to allow for flexible data retrieval.

- **Authors:** This table stores information about the authors.
- **Books:** This table contains information about the books and links to the Authors table using a **foreign key**.
- **Sales:** This table records sales transactions and links to the Books table.

Tables with Constraints

The following constraints were applied to maintain data integrity:

- **PRIMARY KEY:** Used on the ID columns to uniquely identify each row in the Authors and Books tables.
- **FOREIGN KEY:** Links the Books table to the Authors table.
- **NOT NULL:** Ensures required fields, such as names and titles, are never empty.
- **UNIQUE:** Ensures each author name is unique.

Queries for Creating Tables and Inserting Data

Here are the SQL queries we used to create the tables and populate them with sample data.

Creating the tables:

```
CREATE TABLE Authors (
   author_id INT PRIMARY KEY,
   first_name VARCHAR(50) NOT NULL,
   last_name VARCHAR(50) NOT NULL,
   author_name VARCHAR(100) UNIQUE NOT NULL
);

CREATE TABLE Books (
   book_id INT PRIMARY KEY,
   title VARCHAR(200) NOT NULL,
   publication_year INT,
   author_id INT,
   FOREIGN KEY (author_id) REFERENCES Authors(author_id)
);

CREATE TABLE Sales (
   sale_id INT PRIMARY KEY,
   book_id INT,
   sale_date DATE,
   price DECIMAL(5, 2) NOT NULL,
   FOREIGN KEY (book_id) REFERENCES Books(book_id)
);
```

Inserting data:

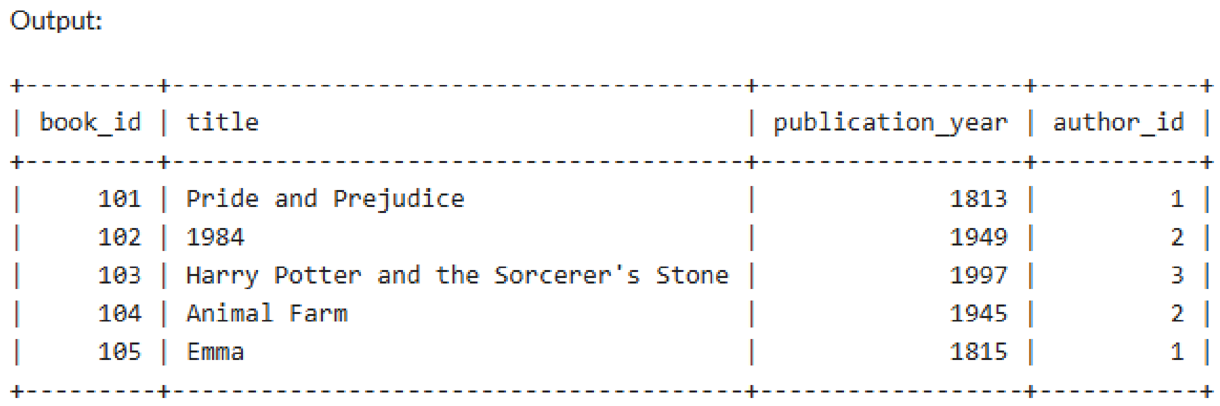
```
-- Insert data into Authors
INSERT INTO Authors (author_id, first_name, last_name, author_name) VALUES
(1, 'Jane', 'Austen', 'Jane Austen'),
(2, 'George', 'Orwell', 'George Orwell'),
(3, 'J.K.', 'Rowling', 'J.K. Rowling');

-- Insert data into Books
INSERT INTO Books (book_id, title, publication_year, author_id) VALUES
(101, 'Pride and Prejudice', 1813, 1),
(102, '1984', 1949, 2),
(103, 'Harry Potter and the Sorcerer''s Stone', 1997, 3),
(104, 'Animal Farm', 1945, 2),
(105, 'Emma', 1815, 1);

-- Insert data into Sales
INSERT INTO Sales (sale_id, book_id, sale_date, price) VALUES
(501, 101, '2025-09-01', 12.99),
(502, 102, '2025-09-02', 15.50),
(503, 104, '2025-09-02', 10.00),
(504, 103, '2025-09-03', 25.75),
(505, 101, '2025-09-04', 12.99);
```

Screenshot of each table after data insertion:

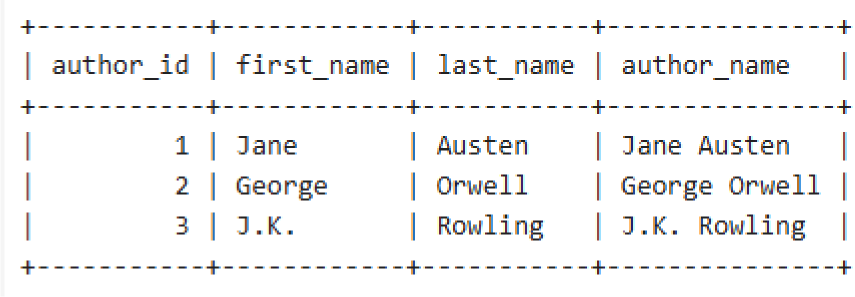
```
SELECT * FROM Books;
```



Output:

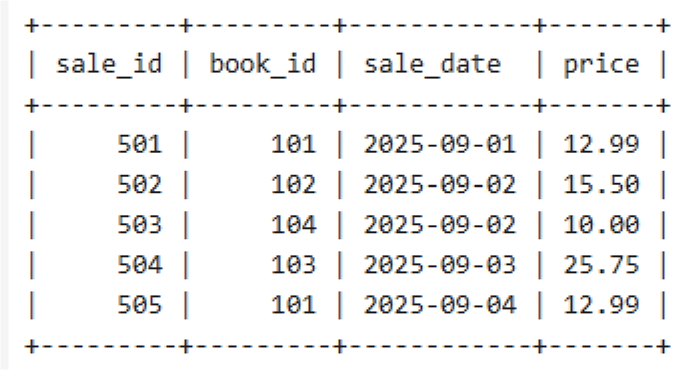
```
+---------+--------------------------------------+-----------------+-----------+
| book_id | title                                | publication_year | author_id |
+---------+--------------------------------------+-----------------+-----------+
|     101 | Pride and Prejudice                  |            1813 |         1 |
|     102 | 1984                                 |            1949 |         2 |
|     103 | Harry Potter and the Sorcerer's Stone |            1997 |         3 |
|     104 | Animal Farm                          |            1945 |         2 |
|     105 | Emma                                 |            1815 |         1 |
+---------+--------------------------------------+-----------------+-----------+
```

SELECT * FROM Authors;



```
+-----------+------------+-----------+---------------+
| author_id | first_name | last_name | author_name   |
+-----------+------------+-----------+---------------+
|         1 | Jane       | Austen    | Jane Austen   |
|         2 | George     | Orwell    | George Orwell |
|         3 | J.K.       | Rowling   | J.K. Rowling  |
+-----------+------------+-----------+---------------+
```

SELECT * FROM Sales;



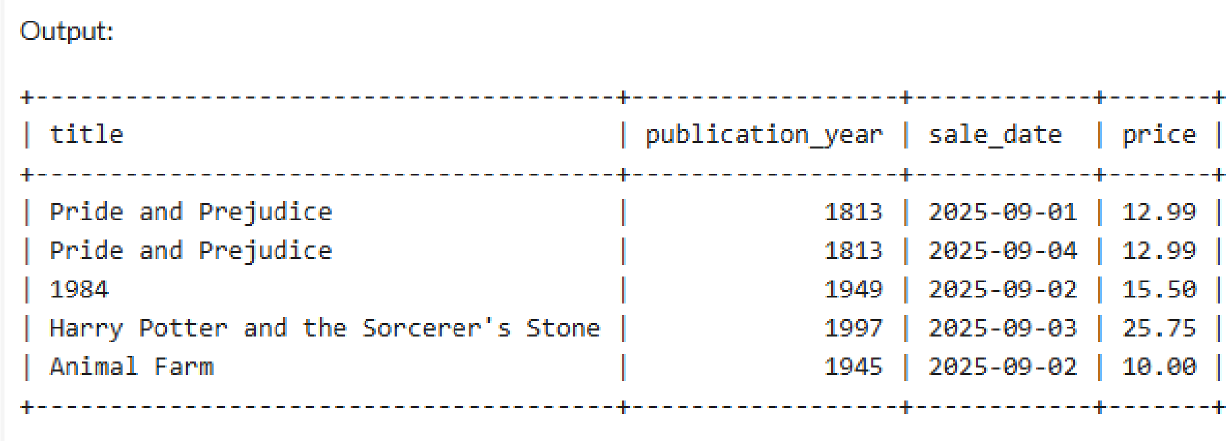
```
+---------+---------+------------+-------+
| sale_id | book_id | sale_date  | price |
+---------+---------+------------+-------+
|     501 |     101 | 2025-09-01 | 12.99 |
|     502 |     102 | 2025-09-02 | 15.50 |
|     503 |     104 | 2025-09-02 | 10.00 |
|     504 |     103 | 2025-09-03 | 25.75 |
|     505 |     101 | 2025-09-04 | 12.99 |
+---------+---------+------------+-------+
```

2. INNER, LEFT, RIGHT, FULL Joins

We performed four different types of joins to demonstrate how data from multiple tables can be combined.

INNER JOIN: This query returns rows that have matching values in both tables.

```
SELECT
    b.title,
    b.publication_year,
    s.sale_date,
    s.price
FROM
    Books b
INNER JOIN
    Sales s ON b.book_id = s.book_id;
```

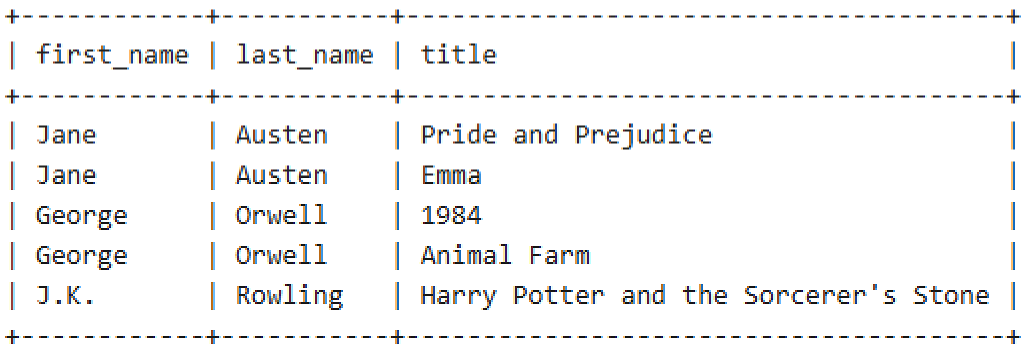


Output:

```
+--------------------------------------+------------------+------------+-------+
| title                                | publication_year | sale_date  | price |
+--------------------------------------+------------------+------------+-------+
| Pride and Prejudice                  |             1813 | 2025-09-01 | 12.99 |
| Pride and Prejudice                  |             1813 | 2025-09-04 | 12.99 |
| 1984                                 |             1949 | 2025-09-02 | 15.50 |
| Harry Potter and the Sorcerer's Stone |             1997 | 2025-09-03 | 25.75 |
| Animal Farm                          |             1945 | 2025-09-02 | 10.00 |
+--------------------------------------+------------------+------------+-------+
```

LEFT JOIN: This query returns all records from the left table, and the matched records from the right table. The result is NULL from the right side if there is no match.

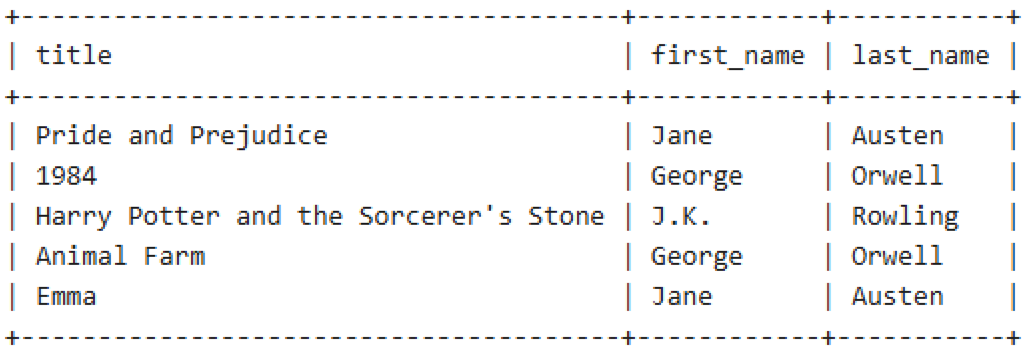
```
SELECT
    a.first_name,
    a.last_name,
    b.title
FROM
    Authors a
LEFT JOIN
    Books b ON a.author_id = b.author_id;
```



```
+------------+-----------+---------------------------------------+
| first_name | last_name | title                                 |
+------------+-----------+---------------------------------------+
| Jane       | Austen    | Pride and Prejudice                   |
| Jane       | Austen    | Emma                                  |
| George     | Orwell    | 1984                                  |
| George     | Orwell    | Animal Farm                           |
| J.K.       | Rowling   | Harry Potter and the Sorcerer's Stone |
+------------+-----------+---------------------------------------+
```

RIGHT JOIN: This query returns all records from the right table, and the matched records from the left table.

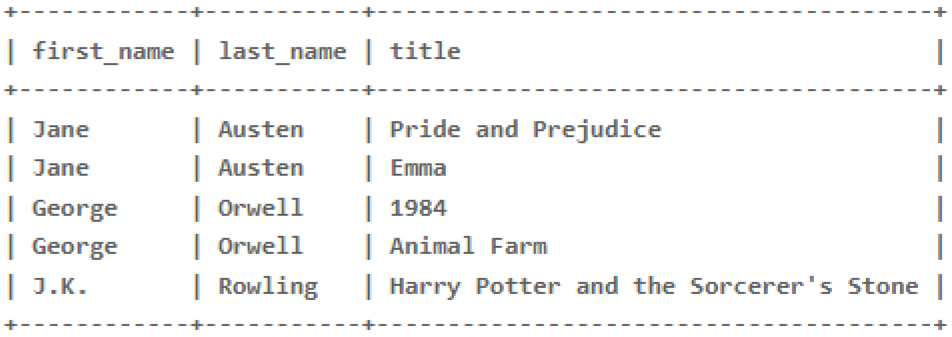
```
SELECT
    b.title,
    a.first_name,
    a.last_name
FROM
    Authors a
RIGHT JOIN
    Books b ON a.author_id = b.author_id;
```



```
+---------------------------------------+------------+-----------+
| title                                 | first_name | last_name |
+---------------------------------------+------------+-----------+
| Pride and Prejudice                   | Jane       | Austen    |
| 1984                                  | George     | Orwell    |
| Harry Potter and the Sorcerer's Stone | J.K.       | Rowling   |
| Animal Farm                           | George     | Orwell    |
| Emma                                  | Jane       | Austen    |
+---------------------------------------+------------+-----------+
```

FULL JOIN: This query returns all records when there is a match in either the left or the right table records.

```
SELECT
    a.first_name,
    a.last_name,
    b.title
FROM
    Authors a
FULL JOIN
    Books b ON a.author_id = b.author_id;
```

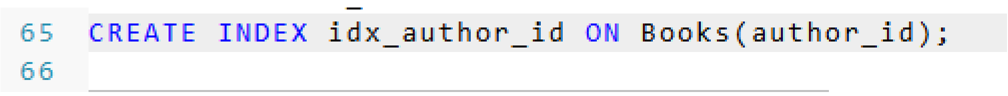


```
+------------+-----------+-------------------------------------+
| first_name | last_name | title                               |
+------------+-----------+-------------------------------------+
| Jane       | Austen    | Pride and Prejudice                 |
| Jane       | Austen    | Emma                                |
| George     | Orwell    | 1984                                |
| George     | Orwell    | Animal Farm                         |
| J.K.       | Rowling   | Harry Potter and the Sorcerer's Stone |
+------------+-----------+-------------------------------------+
```

3. Create an Index

We created an index on the author_id column in the Books table. This is a **foreign key** that is frequently used in joins, so creating an index on it will help to **optimize query performance** and speed up data retrieval.

CREATE INDEX idx_author_id ON Books(author_id);



```
CREATE INDEX idx_author_id ON Books(author_id);
```

And the index was created

4. Create a View

A view is a virtual table that is based on the result-set of a SQL query. We created a view to simplify a complex query that joins Authors and Books tables. This allows us to easily query for books and their authors without writing the full join query every time.

```
CREATE VIEW vw_BooksAndAuthors AS
SELECT
    b.title,
    a.author_name
FROM
    Books b
INNER JOIN
    Authors a ON b.author_id = a.author_id;
```

And the view was created

5. Conclusion

Through this project, we successfully designed and implemented a relational database for a bookstore. We demonstrated our understanding of database design principles, including the use of various constraints, and how to query data efficiently using different types of joins. We also learned how to create indexes and views to improve database performance and simplify data access, which are essential skills for database management and development.