Name:

- DUKUNDIMANA Clovis ID: 27122
- KIRENGA Daniel ID: 27683
- TETA Huguette ID: 28982

Database development with PL/SQL **Group**: D | **Class Quiz(Group Work)**

---

1. Create Tables with Constraints

We'll Create three tables: **Authors, Books** and **Borrowings**

```
-- Authors Table
CREATE TABLE Authors (
    AuthorID INT AUTO_INCREMENT PRIMARY KEY,
    AuthorName VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE
);

-- Books Table
CREATE TABLE Books (
    BookID INT AUTO_INCREMENT PRIMARY KEY,
    Title VARCHAR(150) NOT NULL,
    Genre VARCHAR(50),
    AuthorID INT,
    CONSTRAINT fk_author FOREIGN KEY (AuthorID) REFERENCES
Authors(AuthorID)
);

-- Borrowings Table
CREATE TABLE Borrowings (
    BorrowID INT AUTO_INCREMENT PRIMARY KEY,
    BookID INT NOT NULL,
    BorrowerName VARCHAR(100) NOT NULL,
    BorrowDate DATE NOT NULL,
    ReturnDate DATE,
    CONSTRAINT fk_book FOREIGN KEY (BookID) REFERENCES Books(BookID)
);
```

Constraints applied:

- **Primary Key:** AuthorID, BookID, BorrowID
- **Foreign Key:** Linking **Books** with **Author** and **Borrowings** with **Books.**
- **Unique:** Email in Author
- **Not Null:** AuthorName, Title, BookID, BorrowerName, BorrowDate.

2. Insert Sample Data

```
-- Insert Authors
INSERT INTO Authors (AuthorName, Email) VALUES
('George Orwell', 'orwell@example.com'),
('J.K. Rowling', 'rowling@example.com'),
('Chinua Achebe', 'achebe@example.com');

-- Insert Books
INSERT INTO Books (Title, Genre, AuthorID) VALUES
('1984', 'Dystopian', 1),
('Animal Farm', 'Satire', 1),
('Harry Potter', 'Fantasy', 2),
('Things Fall Apart', 'Historical Fiction', 3);

-- Insert Borrowings
INSERT INTO Borrowings (BookID, BorrowerName, BorrowDate, ReturnDate)
VALUES
(1, 'Alice', '2025-09-01', '2025-09-10'),
(2, 'Bob', '2025-09-05', NULL),
(3, 'Charlie', '2025-09-08', '2025-09-15');
```

3. Perform Different Joins

```
-- INNER JOIN: Books with their authors
SELECT b.Title, a.AuthorName, b.Genre
FROM Books b
INNER JOIN Authors a ON b.AuthorID = a.AuthorID;

-- LEFT JOIN: All books, even if never borrowed
SELECT b.Title, br.BorrowerName, br.BorrowDate
FROM Books b
LEFT JOIN Borrowings br ON b.BookID = br.BookID;

-- RIGHT JOIN: All borrowings, even if book details missing
SELECT b.Title, br.BorrowerName, br.BorrowDate
FROM Books b
RIGHT JOIN Borrowings br ON b.BookID = br.BookID;

-- FULL OUTER JOIN (simulate in MySQL using UNION)
SELECT b.Title, br.BorrowerName, br.BorrowDate
FROM Books b
LEFT JOIN Borrowings br ON b.BookID = br.BookID
UNION
SELECT b.Title, br.BorrowerName, br.BorrowDate
FROM Books b
RIGHT JOIN Borrowings br ON b.BookID = br.BookID;
```

4. Create an Index to optimize performance

```
-- Index on BookID for faster borrowing lookups
CREATE INDEX idx_borrowings_bookid ON Borrowings(BookID);
```

5. Create a View

```
CREATE VIEW vw_BookBorrowings AS
SELECT b.Title, a.AuthorName, br.BorrowerName, br.BorrowDate,
br.ReturnDate
FROM Books b
INNER JOIN Authors a ON b.AuthorID = a.AuthorID
LEFT JOIN Borrowings br ON b.BookID = br.BookID;
```

Now, instead of writing big joins, we can just do:

```
SELECT * FROM vw_BookBorrowings;
```

6. Simple Report

What we did:

- Created Authors, Books, and Borrowings tables with primary/foreign keys and constraints.
- Inserted sample data for authors, books, and borrowing transactions.
- Used INNER, LEFT, RIGHT, FULL OUTER JOIN queries to explore relationships.
- Created an index on Borrowings.BookID to optimize queries.
- Built a view vw_BookBorrowings to easily fetch combined data.

Example Output (from View):

Title	AuthorName	BorrowerName	BorrowDate	ReturnDate
1984	George Orwell	Alice	2025-09-01	2025-09-10
Animal Farm	George Orwell	Bob	2025-09-05	NULL
Harry Potter	J.K. Rowling	Charlie	2025-09-08	2025-09-15
Things Fall Apart	Chinua Achebe	NULL	NULL	NULL