Script MySQL LibraTech

```
-- Mulai transaksi
START TRANSACTION;

-- 1. CREATE DATABASE
CREATE DATABASE IF NOT EXISTS Perpustakaan;
USE Perpustakaan;

-- 2. CREATE TABLES
-- Tabel Anggota
CREATE TABLE IF NOT EXISTS Anggota (
    id_anggota INT AUTO_INCREMENT PRIMARY KEY,
    nama VARCHAR(100) NOT NULL,
    alamat TEXT,
    telepon VARCHAR(15)
);

-- Tabel Kategori
CREATE TABLE IF NOT EXISTS Kategori (
    id_kategori INT AUTO_INCREMENT PRIMARY KEY,
    nama_kategori VARCHAR(50) NOT NULL
);

-- Tabel Buku
CREATE TABLE IF NOT EXISTS Buku (
    id_buku INT AUTO_INCREMENT PRIMARY KEY,
    judul VARCHAR(100) NOT NULL,
    penulis VARCHAR(100),
    id_kategori INT,
    FOREIGN KEY (id_kategori) REFERENCES Kategori(id_kategori)
);

-- Tabel Peminjaman
CREATE TABLE IF NOT EXISTS Peminjaman (
    id_peminjaman INT AUTO_INCREMENT PRIMARY KEY,
    id_anggota INT,
    id_buku INT,
    tanggal_pinjam DATE NOT NULL,
    tanggal_kembali DATE,
    FOREIGN KEY (id_anggota) REFERENCES Anggota(id_anggota),
    FOREIGN KEY (id_buku) REFERENCES Buku(id_buku)
);

-- 3. CREATE VIEW
-- View untuk melihat daftar peminjaman lengkap dengan nama anggota dan judul buku
CREATE VIEW ViewPeminjaman AS
SELECT
    Peminjaman.id_peminjaman,
    Anggota.nama AS nama_anggota,
```

```
    Buku.judul AS judul_buku,
    Peminjaman.tanggal_pinjam,
    Peminjaman.tanggal_kembali
FROM Peminjaman
JOIN Anggota ON Peminjaman.id_anggota = Anggota.id_anggota
JOIN Buku ON Peminjaman.id_buku = Buku.id_buku;

-- 4. CREATE STORED PROCEDURE
-- Stored Procedure untuk menambahkan data peminjaman
DELIMITER //
CREATE PROCEDURE TambahPeminjaman(
    IN p_id_anggota INT,
    IN p_id_buku INT,
    IN p_tanggal_pinjam DATE,
    IN p_tanggal_kembali DATE
)
BEGIN
    -- Error handling: Cek apakah buku sudah dipinjam
    DECLARE buku_dipinjam INT;
    SELECT COUNT(*) INTO buku_dipinjam
    FROM Peminjaman
    WHERE id_buku = p_id_buku AND tanggal_kembali IS NULL;

    IF buku_dipinjam > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Buku ini sedang dipinjam dan belum dikembalikan.';
    ELSE
        -- Insert data peminjaman
        INSERT INTO Peminjaman (id_anggota, id_buku, tanggal_pinjam,
tanggal_kembali)
        VALUES (p_id_anggota, p_id_buku, p_tanggal_pinjam, p_tanggal_kembali);
    END IF;
END;
//
DELIMITER ;

-- 5. CREATE CURSOR
-- Procedure untuk menampilkan semua buku berdasarkan kategori tertentu
DELIMITER //
CREATE PROCEDURE LihatBukuPerKategori(IN p_id_kategori INT)
BEGIN
    DECLARE selesai INT DEFAULT 0;
    DECLARE buku_judul VARCHAR(100);
    DECLARE buku_cursor CURSOR FOR
        SELECT judul FROM Buku WHERE id_kategori = p_id_kategori;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET selesai = 1;

    OPEN buku_cursor;

    buku_loop: LOOP
```

```
        FETCH buku_cursor INTO buku_judul;
        IF selesai THEN
            LEAVE buku_loop;
        END IF;
        SELECT buku_judul AS "Judul Buku";
    END LOOP;

    CLOSE buku_cursor;
END;
//
DELIMITER ;

-- 6. CREATE TRIGGER
-- Trigger untuk mencatat log setiap kali ada peminjaman baru
CREATE TABLE IF NOT EXISTS LogPeminjaman (
    id_log INT AUTO_INCREMENT PRIMARY KEY,
    id_peminjaman INT,
    waktu_log TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

DELIMITER //
CREATE TRIGGER AfterInsertPeminjaman
AFTER INSERT ON Peminjaman
FOR EACH ROW
BEGIN
    INSERT INTO LogPeminjaman (id_peminjaman)
    VALUES (NEW.id_peminjaman);
END;
//
DELIMITER ;

-- Commit transaksi
COMMIT;
```

```
-- Tabel Anggota
ALTER TABLE Anggota
ADD COLUMN created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
ADD COLUMN updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP;

-- Tabel Kategori
ALTER TABLE Kategori
ADD COLUMN created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
ADD COLUMN updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP;

-- Tabel Buku
ALTER TABLE Buku
ADD COLUMN created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
ADD COLUMN updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP;
```