```
1. CREATE TABLE Admin (
  AdminId INT UNSIGNED NOT NULL AUTO_INCREMENT,
  UserName VARCHAR(50) NOT NULL,
  Email VARCHAR(100) NOT NULL,
  Password VARCHAR(50) NOT NULL,
  Status ENUM('active','inactive') NOT NULL DEFAULT 'inactive',
  NewBookReq INT DEFAULT 0 CHECK (NewBookReq >=0),
  NewUserReq  INT DEFAULT 0 CHECK (NewUserReq>=0),
   PRIMARY KEY (AdminId)
) AUTO_INCREMENT=101;

2. CREATE TABLE Users (
  UserId INT UNSIGNED NOT NULL AUTO_INCREMENT,
  Name VARCHAR(50) NOT NULL,
  Password VARCHAR(50) NOT NULL,
  Email VARCHAR(100) NOT NULL,
  Hostel_Name VARCHAR(50) NOT NULL,
  Room_No VARCHAR(20) NOT NULL,
  Phone_No INT NOT NULL,
  Status ENUM('active','inactive') NOT NULL DEFAULT 'inactive',
  Created_By ENUM('user','admin') NOT NULL DEFAULT 'user',
  Verified_BY INT UNSIGNED  NULL,
  BorrowRequest INT DEFAULT 0 CHECK (BorrowRequest>=0),
  ExchangeRequest INT DEFAULT 0 CHECK (ExchangeRequest >=0),
  PRIMARY KEY (UserId),
  CONSTRAINT Created_by FOREIGN KEY (Verified_BY)
    REFERENCES Admin(AdminId)
) AUTO_INCREMENT=201;

DELIMITER $$
CREATE TRIGGER notify_admins1
AFTER INSERT ON `Users`
FOR EACH ROW
BEGIN
    IF NEW.Verified_BY IS NULL THEN
        UPDATE Admin SET NewUserReq = NewUserReq+ 1 ;
    END IF;
END$$
DELIMITER ;


DELIMITER $$
CREATE TRIGGER notify_admins2
AFTER UPDATE ON `Users`
FOR EACH ROW
BEGIN
    IF NEW.Verified_BY IS NOT NULL AND OLD.Verified_BY IS NULL THEN
        UPDATE Admin SET NewUserReq = NewUserReq - 1;
```

```
    END IF;
END$$
DELIMITER ;


3. CREATE TABLE Add_Book_Request (
    Req_no INT UNSIGNED NOT NULL AUTO_INCREMENT,
    Requested_by INT UNSIGNED NOT NULL,
    Added_by INT UNSIGNED NULL,
    Book_Title VARCHAR(100) NOT NULL,
    ISBN_NO VARCHAR(50) NOT NULL,
    Book_type VARCHAR(50) NOT NULL,
    Book_Rating DECIMAL(3, 1) NOT NULL CHECK (Book_Rating >= 1 AND Book_Rating
<= 5),
    Publication_Year INT NOT NULL,
    Book_Authors VARCHAR(100) NOT NULL,
    Book_condition ENUM('New', 'Old', 'Good') NOT NULL DEFAULT 'Good',
    PRIMARY KEY (Req_no),
    FOREIGN KEY (Requested_by) REFERENCES Users(UserId),
    FOREIGN KEY (Added_by ) REFERENCES Admin(AdminId)
) AUTO_INCREMENT=1;



4.WHERE AdminId = (SELECT Verified_BY FROM Users WHERE UserId =
NEW.Requested_by)

DELIMITER $$
CREATE TRIGGER notify_admins3
AFTER INSERT ON `Add_Book_Request`
FOR EACH ROW
BEGIN
    IF NEW.Added_by IS NULL THEN
        UPDATE Admin SET NewBookReq = NewBookReq+ 1 ;
    END IF;
END$$
DELIMITER ;



DELIMITER $$
CREATE TRIGGER notify_admins4
AFTER UPDATE ON `Add_Book_Request`
FOR EACH ROW
BEGIN
    IF NEW.Added_by IS NOT NULL AND OLD.Added_by IS NULL THEN
        UPDATE Admin SET NewBookReq = NewBookReq - 1;
    END IF;
END$$
```

```
DELIMITER ;
```

5 .

```
CREATE TABLE Wish_List (
List_Id INT UNSIGNED NOT NULL AUTO_INCREMENT,
Book_Title VARCHAR(100) NOT NULL,
Publication_Year INT NOT NULL,
Wishers_Id INT UNSIGNED NOT NULL,
PRIMARY KEY (List_Id),
FOREIGN KEY (Wishers_Id) REFERENCES Users(UserId)
) AUTO_INCREMENT=1;
```

6.

```
CREATE TABLE Book (
Book_Id INT UNSIGNED NOT NULL AUTO_INCREMENT,
Book_Owner  INT UNSIGNED NOT NULL,
Book_Title VARCHAR(100) NOT NULL,
ISBN_NO VARCHAR(50) NOT NULL,
Publication_Year INT NOT NULL,
Book_Authors VARCHAR(100) NOT NULL,
Book_type VARCHAR(50) NOT NULL,
Book_Rating DECIMAL(3, 1) NOT NULL CHECK (Book_Rating >= 1 AND Book_Rating <=
5),
Book_condition ENUM('New','Old','Good') NOT NULL DEFAULT 'Good',
Availability ENUM('Exchanged','Borrowed','Available') NOT NULL DEFAULT 'Available',
BookAdded_by INT UNSIGNED NOT NULL,
Action ENUM('Add','Delete','Update') NOT NULL DEFAULT 'Add',
PRIMARY KEY (Book_Id),
FOREIGN KEY (BookAdded_by) REFERENCES Admin(AdminId),
FOREIGN KEY (Book_Owner) REFERENCES Add_Book_Request(Requested_by)
) AUTO_INCREMENT=1;
```

and fix the issue by rewiring the update_added_by tigger sql command

```
DELIMITER $$
CREATE TRIGGER update_added_by
AFTER INSERT ON Book
FOR EACH ROW
BEGIN
  IF NEW.BookAdded_by IS NOT NULL THEN
    UPDATE Add_Book_Request
    SET Added_by = NEW.BookAdded_by.AdminId
```

```
    WHERE Requested_by = NEW.Book_Owner AND ISBN_NO = NEW.ISBN_NO;
  END IF;
END$$
DELIMITER ;
```

7.

```
CREATE TABLE Borrow_request (
Borrow_No INT UNSIGNED NOT NULL AUTO_INCREMENT,
Owner_Id INT UNSIGNED NOT NULL,
Borrower_Id INT UNSIGNED NOT NULL,
Book_Id INT UNSIGNED NOT NULL,
Start_Date DATE NOT NULL DEFAULT (CURRENT_DATE),
End_Date DATE NOT NULL,
Status ENUM('Pending','Accepted','Rejected') NOT NULL DEFAULT 'Pending',
Comment VARCHAR(250),
PRIMARY KEY (Borrow_No),
FOREIGN KEY (Book_Id) REFERENCES Book(Book_Id),
FOREIGN KEY (Owner_Id) REFERENCES Users(UserId),
FOREIGN KEY (Borrower_Id) REFERENCES Users(UserId)
) AUTO_INCREMENT=1;
```

```
DELIMITER $$
CREATE TRIGGER update_borrow_req
AFTER INSERT ON `Borrow_request`
FOR EACH ROW
BEGIN
    UPDATE Users
   SET BorrowRequest = BorrowRequest + 1
   WHERE UserId = NEW.Owner_Id;
END$$
DELIMITER ;
```

```
DELIMITER $$
CREATE TRIGGER delete_borrow_req
AFTER DELETE ON Borrow_request
FOR EACH ROW
BEGIN
```

```
        UPDATE Users
   SET BorrowRequest = BorrowRequest - 1
   WHERE UserId = OLD.Owner_Id;
END$$
DELIMITER ;
```

8.

```
CREATE TABLE Exchange_request (
Exchange_No INT UNSIGNED NOT NULL AUTO_INCREMENT,
User1_Id INT UNSIGNED NOT NULL,
User2_Id INT UNSIGNED NOT NULL,
Book1_Id INT UNSIGNED NOT NULL,
Book2_Id INT UNSIGNED NOT NULL,
Request_Date DATE NOT NULL,
Exchange_Date DATE NOT NULL,
User1_Status ENUM('Pending', 'Accepted','Rejected') NOT NULL DEFAULT 'Pending',
User2_Status ENUM('Pending', 'Accepted','Rejected') NOT NULL DEFAULT 'Pending',
Comment VARCHAR(250),
PRIMARY KEY (Exchange_No),
FOREIGN KEY (Book1_Id) REFERENCES Book(Book_Id),
FOREIGN KEY (Book2_Id) REFERENCES Book(Book_Id),
FOREIGN KEY (User1_Id) REFERENCES Users(UserId),
FOREIGN KEY (User1_Id) REFERENCES Users(UserId)
) AUTO_INCREMENT=1;

DELIMITER $$
CREATE TRIGGER update_exchage _req
AFTER INSERT ON Exchange_request
FOR EACH ROW
BEGIN
    UPDATE Users
   SET ExchangeRequest = ExchangeRequest + 1
   WHERE UserId = NEW.User1_Id OR UserId = NEW.User2_Id ;
END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER delete_exchage _req
AFTER DELETE ON Exchange_request
FOR EACH ROW
BEGIN
    UPDATE Users
```

```
    SET ExchangeRequest = ExchangeRequest - 1
    WHERE UserId = NEW.User1_Id AND UserId = NEW.User2_Id ;
END IF;
END$$
DELIMITER ;
```