WEEK 12

1. SELECT \*

FROM LibraryBooks

WHERE Copies < 3;

1. SELECT lb.\*

FROM LibraryBooks lb

JOIN IssuedBooks ib ON lb.BookId = ib.BookId

WHERE ib.MemberId = 101;

1. SELECT ib.\*, lb.Title, lb.Author

FROM IssuedBooks ib

JOIN LibraryBooks lb ON ib.BookId = lb.BookId

WHERE ib.ReturnDate < SYSDATE AND ib.Status <> 'Returned';

1. SELECT \*

FROM LibraryBooks

WHERE BookId NOT IN (SELECT DISTINCT BookId FROM IssuedBooks);

1. SELECT \*

FROM Members

WHERE MemberId NOT IN (SELECT DISTINCT MemberId FROM IssuedBooks);

1. SELECT BookId, Title

FROM LibraryBooks

WHERE BookId = (

SELECT BookId

FROM IssuedBooks

GROUP BY BookId

HAVING COUNT(\*) = (SELECT MAX(COUNT(\*)) FROM IssuedBooks GROUP BY BookId)

);

1. CREATE OR REPLACE PROCEDURE IssueBook(

p\_MemberId IN NUMBER,

p\_BookId IN NUMBER

) IS

BEGIN

UPDATE LibraryBooks

SET Copies = Copies - 1

WHERE BookId = p\_BookId AND Copies > 0;

**[ IF SQL%ROWCOUNT = 0 THEN**

**RAISE\_APPLICATION\_ERROR(-20002, 'No copies available.');**

**END IF;]**

INSERT INTO IssuedBooks (IssueId, MemberId, BookId, IssueDate, ReturnDate, Status)

VALUES (1001, p\_MemberId, p\_BookId, SYSDATE, SYSDATE + 14, 'Issued');

END;

/

8. CREATE OR REPLACE FUNCTION CountBorrowedBooks RETURN NUMBER IS

v\_Count NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_Count FROM IssuedBooks;

RETURN v\_Count;

END;

/

1. CREATE OR REPLACE TRIGGER LogBookIssue

AFTER INSERT ON IssuedBooks

FOR EACH ROW

BEGIN

INSERT INTO BookIssueLog (IssueId, MemberId, BookId, IssueDate)

VALUES (:NEW.IssueId, :NEW.MemberId, :NEW.BookId, :NEW.IssueDate);

END;

/

10. CREATE OR REPLACE TRIGGER PreventNegativeCopies

BEFORE UPDATE ON LibraryBooks

FOR EACH ROW

BEGIN

IF :NEW.Copies < 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Copies cannot be negative.');

END IF;

END;

/