**FOUNDICU Package Documentation**

**Introduction**

The FOUNDICU package is designed to manage library loans and reservations in the FSDB (Flying Squirrel Database) system. Based on the SQL scripts provided, we can see that the database supports a mobile library system with buses that stop at various municipalities, allowing users to borrow and return books.

The package contains three main procedures:

1. Insert Loan Procedure
2. Insert Reservation Procedure
3. Record Books Returning Procedure

Each procedure interacts with multiple tables in the database to ensure data integrity and proper functionality of the library system.

**Package Structure**

CREATE OR REPLACE PACKAGE foundicu AS

PROCEDURE insert\_loan(p\_signature IN copies.signature%TYPE);

PROCEDURE insert\_reservation(p\_isbn IN editions.isbn%TYPE, p\_date IN DATE);

PROCEDURE record\_return(p\_signature IN copies.signature%TYPE);

END foundicu;

/

CREATE OR REPLACE PACKAGE BODY foundicu AS

-- Implementation of procedures

END foundicu;

/

**1. Insert Loan Procedure**

**Design**

**Inputs:**

* p\_signature: The signature of the copy to be loaned

**Outputs:**

* Success message when loan is processed
* Error message when loan cannot be processed, with reason

**Logic:**

1. Check if the current user exists in the system
2. Check if there is a reservation (hold) for this copy for the current user as of today
   * If yes, convert the reservation to a loan
3. If no reservation exists:
   * Check if the copy is available for two weeks
   * Check if the user hasn't reached their loan limit
   * Check if the user isn't currently banned
4. If conditions are met, insert a new loan record
5. If conditions aren't met, report the problem

**Auxiliary Elements:**

* Query to check if user exists
* Query to check for existing reservation
* Query to check copy availability
* Query to check user's current loan count
* Query to check if user is banned

**Implementation**

PROCEDURE insert\_loan(p\_signature IN copies.signature%TYPE) IS

v\_user\_id users.user\_id%TYPE;

v\_user\_exists NUMBER;

v\_has\_reservation NUMBER;

v\_town stops.town%TYPE;

v\_province stops.province%TYPE;

v\_stopdate DATE;

v\_copy\_available NUMBER;

v\_loan\_count NUMBER;

v\_is\_banned NUMBER;

v\_max\_loans NUMBER := 5; -- Assuming 5 is the maximum number of loans allowed

BEGIN

-- Get current user ID (assuming it's stored in a context or session)

v\_user\_id := SYS\_CONTEXT('USER\_CONTEXT', 'USER\_ID');

-- Check if user exists

SELECT COUNT(\*) INTO v\_user\_exists

FROM users

WHERE user\_id = v\_user\_id;

IF v\_user\_exists = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Error: User does not exist in the system.');

RETURN;

END IF;

-- Check if there is a reservation for this copy for the current user as of today

SELECT COUNT(\*), MAX(town), MAX(province), MAX(stopdate)

INTO v\_has\_reservation, v\_town, v\_province, v\_stopdate

FROM loans

WHERE signature = p\_signature

AND user\_id = v\_user\_id

AND TRUNC(SYSDATE) = TRUNC(stopdate)

AND return IS NULL

AND type = 'R'; -- Assuming 'R' is the code for reservation

IF v\_has\_reservation > 0 THEN

-- Convert reservation to loan

UPDATE loans

SET type = 'L', -- Assuming 'L' is the code for loan

time = 0 -- Reset time counter

WHERE signature = p\_signature

AND user\_id = v\_user\_id

AND TRUNC(stopdate) = TRUNC(SYSDATE)

AND type = 'R';

DBMS\_OUTPUT.PUT\_LINE('Reservation converted to loan successfully.');

ELSE

-- Check if copy is available for two weeks

SELECT COUNT(\*)

INTO v\_copy\_available

FROM copies

WHERE signature = p\_signature

AND deregistered IS NULL

AND NOT EXISTS (

SELECT 1

FROM loans

WHERE signature = p\_signature

AND return IS NULL

AND TRUNC(stopdate) BETWEEN TRUNC(SYSDATE) AND TRUNC(SYSDATE) + 14

);

IF v\_copy\_available = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Copy is not available for the next two weeks.');

RETURN;

END IF;

-- Check if user has reached loan limit

SELECT COUNT(\*)

INTO v\_loan\_count

FROM loans

WHERE user\_id = v\_user\_id

AND type = 'L'

AND return IS NULL;

IF v\_loan\_count >= v\_max\_loans THEN

DBMS\_OUTPUT.PUT\_LINE('Error: User has reached the maximum number of loans.');

RETURN;

END IF;

-- Check if user is banned

SELECT COUNT(\*)

INTO v\_is\_banned

FROM users

WHERE user\_id = v\_user\_id

AND ban\_up2 IS NOT NULL

AND ban\_up2 >= SYSDATE;

IF v\_is\_banned > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Error: User is currently banned from borrowing.');

RETURN;

END IF;

-- Find current service (town, province, date)

BEGIN

SELECT town, province, taskdate

INTO v\_town, v\_province, v\_stopdate

FROM services

WHERE TRUNC(taskdate) = TRUNC(SYSDATE)

AND ROWNUM = 1;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: No service available today.');

RETURN;

END;

-- Insert new loan

INSERT INTO loans (signature, user\_id, stopdate, town, province, type, time, return)

VALUES (p\_signature, v\_user\_id, v\_stopdate, v\_town, v\_province, 'L', 0, NULL);

DBMS\_OUTPUT.PUT\_LINE('Loan created successfully.');

END IF;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END insert\_loan;

**Tests**

-- Test 1: User doesn't exist

BEGIN

-- Set context to non-existent user

DBMS\_SESSION.SET\_CONTEXT('USER\_CONTEXT', 'USER\_ID', 'NONEXIST1');

foundicu.insert\_loan('A1234');

END;

/

-- Expected output: Error: User does not exist in the system.

-- Test 2: User has a reservation for today

BEGIN

-- Set context to existing user with reservation

DBMS\_SESSION.SET\_CONTEXT('USER\_CONTEXT', 'USER\_ID', 'USER001');

foundicu.insert\_loan('B5678');

END;

/

-- Expected output: Reservation converted to loan successfully.

-- Test 3: Copy not available

BEGIN

-- Set context to existing user

DBMS\_SESSION.SET\_CONTEXT('USER\_CONTEXT', 'USER\_ID', 'USER002');

foundicu.insert\_loan('C9012');

END;

/

-- Expected output: Error: Copy is not available for the next two weeks.

-- Test 4: User reached maximum loans

BEGIN

-- Set context to user with maximum loans

DBMS\_SESSION.SET\_CONTEXT('USER\_CONTEXT', 'USER\_ID', 'USER003');

foundicu.insert\_loan('D3456');

END;

/

-- Expected output: Error: User has reached the maximum number of loans.

-- Test 5: User is banned

BEGIN

-- Set context to banned user

DBMS\_SESSION.SET\_CONTEXT('USER\_CONTEXT', 'USER\_ID', 'USER004');

foundicu.insert\_loan('E7890');

END;

/

-- Expected output: Error: User is currently banned from borrowing.

-- Test 6: Successful loan

BEGIN

-- Set context to eligible user

DBMS\_SESSION.SET\_CONTEXT('USER\_CONTEXT', 'USER\_ID', 'USER005');

foundicu.insert\_loan('F1234');

END;

/

-- Expected output: Loan created successfully.

**2. Insert Reservation Procedure**

**Design**

**Inputs:**

* p\_isbn: The ISBN of the book edition to be reserved
* p\_date: The date for which the reservation is made

**Outputs:**

* Success message when reservation is processed
* Error message when reservation cannot be processed, with reason

**Logic:**

1. Check if the current user exists in the system
2. Check if the user hasn't reached their reservation limit
3. Check if the user isn't currently banned
4. Check for availability of a copy of the requested edition for two weeks from the provided date
5. If conditions are met, place the reservation (hold)
6. If conditions aren't met, report the problem

**Auxiliary Elements:**

* Query to check if user exists
* Query to check user's current reservation count
* Query to check if user is banned
* Query to find available copy of the requested edition
* Query to find service information for the requested date

**Implementation**

PROCEDURE insert\_reservation(p\_isbn IN editions.isbn%TYPE, p\_date IN DATE) IS

v\_user\_id users.user\_id%TYPE;

v\_user\_exists NUMBER;

v\_reservation\_count NUMBER;

v\_is\_banned NUMBER;

v\_max\_reservations NUMBER := 3; -- Assuming 3 is the maximum number of reservations allowed

v\_town stops.town%TYPE;

v\_province stops.province%TYPE;

v\_service\_date DATE;

v\_available\_signature copies.signature%TYPE;

v\_service\_exists NUMBER;

BEGIN

-- Get current user ID (assuming it's stored in a context or session)

v\_user\_id := SYS\_CONTEXT('USER\_CONTEXT', 'USER\_ID');

-- Check if user exists

SELECT COUNT(\*) INTO v\_user\_exists

FROM users

WHERE user\_id = v\_user\_id;

IF v\_user\_exists = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Error: User does not exist in the system.');

RETURN;

END IF;

-- Check if user has reached reservation limit

SELECT COUNT(\*)

INTO v\_reservation\_count

FROM loans

WHERE user\_id = v\_user\_id

AND type = 'R' -- Assuming 'R' is the code for reservation

AND stopdate >= SYSDATE;

IF v\_reservation\_count >= v\_max\_reservations THEN

DBMS\_OUTPUT.PUT\_LINE('Error: User has reached the maximum number of reservations.');

RETURN;

END IF;

-- Check if user is banned

SELECT COUNT(\*)

INTO v\_is\_banned

FROM users

WHERE user\_id = v\_user\_id

AND ban\_up2 IS NOT NULL

AND ban\_up2 >= SYSDATE;

IF v\_is\_banned > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Error: User is currently banned from borrowing.');

RETURN;

END IF;

-- Check if service exists for the specified date

SELECT COUNT(\*)

INTO v\_service\_exists

FROM services

WHERE TRUNC(taskdate) = TRUNC(p\_date);

IF v\_service\_exists = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Error: No service available on the specified date.');

RETURN;

END IF;

-- Find a copy that is available for the next two weeks

BEGIN

SELECT c.signature

INTO v\_available\_signature

FROM copies c

JOIN editions e ON c.isbn = e.isbn

WHERE e.isbn = p\_isbn

AND c.deregistered IS NULL

AND NOT EXISTS (

SELECT 1

FROM loans l

WHERE l.signature = c.signature

AND l.return IS NULL

AND l.stopdate BETWEEN TRUNC(p\_date) AND TRUNC(p\_date) + 14

)

AND ROWNUM = 1;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: No available copy of this edition for the specified date range.');

RETURN;

END;

-- Get service information for the requested date

SELECT town, province, taskdate

INTO v\_town, v\_province, v\_service\_date

FROM services

WHERE TRUNC(taskdate) = TRUNC(p\_date)

AND ROWNUM = 1;

-- Insert reservation

INSERT INTO loans (signature, user\_id, stopdate, town, province, type, time, return)

VALUES (v\_available\_signature, v\_user\_id, v\_service\_date, v\_town, v\_province, 'R', 0, NULL);

DBMS\_OUTPUT.PUT\_LINE('Reservation created successfully for copy ' || v\_available\_signature || ' on ' || TO\_CHAR(p\_date, 'DD-MM-YYYY'));

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END insert\_reservation;

**Tests**

-- Test 1: User doesn't exist

BEGIN

-- Set context to non-existent user

DBMS\_SESSION.SET\_CONTEXT('USER\_CONTEXT', 'USER\_ID', 'NONEXIST1');

foundicu.insert\_reservation('978-3-16-148410-0', SYSDATE + 7);

END;

/

-- Expected output: Error: User does not exist in the system.

-- Test 2: User reached maximum reservations

BEGIN

-- Set context to user with maximum reservations

DBMS\_SESSION.SET\_CONTEXT('USER\_CONTEXT', 'USER\_ID', 'USER006');

foundicu.insert\_reservation('978-3-16-148410-0', SYSDATE + 7);

END;

/

-- Expected output: Error: User has reached the maximum number of reservations.

-- Test 3: User is banned

BEGIN

-- Set context to banned user

DBMS\_SESSION.SET\_CONTEXT('USER\_CONTEXT', 'USER\_ID', 'USER004');

foundicu.insert\_reservation('978-3-16-148410-0', SYSDATE + 7);

END;

/

-- Expected output: Error: User is currently banned from borrowing.

-- Test 4: No service on specified date

BEGIN

-- Set context to eligible user

DBMS\_SESSION.SET\_CONTEXT('USER\_CONTEXT', 'USER\_ID', 'USER007');

foundicu.insert\_reservation('978-3-16-148410-0', TO\_DATE('31-12-2099', 'DD-MM-YYYY'));

END;

/

-- Expected output: Error: No service available on the specified date.

-- Test 5: No available copy

BEGIN

-- Set context to eligible user

DBMS\_SESSION.SET\_CONTEXT('USER\_CONTEXT', 'USER\_ID', 'USER007');

foundicu.insert\_reservation('978-0-00-000000-0', SYSDATE + 7); -- Non-existent or unavailable ISBN

END;

/

-- Expected output: Error: No available copy of this edition for the specified date range.

-- Test 6: Successful reservation

BEGIN

-- Set context to eligible user

DBMS\_SESSION.SET\_CONTEXT('USER\_CONTEXT', 'USER\_ID', 'USER007');

foundicu.insert\_reservation('978-3-16-148410-0', SYSDATE + 7);

END;

/

-- Expected output: Reservation created successfully for copy X#### on DD-MM-YYYY

**3. Record Books Returning Procedure**

**Design**

**Inputs:**

* p\_signature: The signature of the copy being returned

**Outputs:**

* Success message when return is processed
* Error message when return cannot be processed, with reason

**Logic:**

1. Check if the current user has borrowed the book
2. If yes, record the return by updating the loan record
3. If no, report that the user hasn't borrowed this book

**Auxiliary Elements:**

* Query to check if the current user has an active loan for the given copy

**Implementation**

PROCEDURE record\_return(p\_signature IN copies.signature%TYPE) IS

v\_user\_id users.user\_id%TYPE;

v\_loan\_exists NUMBER;

BEGIN

-- Get current user ID (assuming it's stored in a context or session)

v\_user\_id := SYS\_CONTEXT('USER\_CONTEXT', 'USER\_ID');

-- Check if user has borrowed this book

SELECT COUNT(\*)

INTO v\_loan\_exists

FROM loans

WHERE signature = p\_signature

AND user\_id = v\_user\_id

AND type = 'L' -- Assuming 'L' is the code for loan

AND return IS NULL;

IF v\_loan\_exists > 0 THEN

-- Record return

UPDATE loans

SET return = SYSDATE

WHERE signature = p\_signature

AND user\_id = v\_user\_id

AND type = 'L'

AND return IS NULL;

DBMS\_OUTPUT.PUT\_LINE('Book return recorded successfully.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Error: User has not borrowed this book or it has already been returned.');

END IF;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END record\_return;

**Tests**

-- Test 1: User hasn't borrowed the book

BEGIN

-- Set context to user

DBMS\_SESSION.SET\_CONTEXT('USER\_CONTEXT', 'USER\_ID', 'USER008');

foundicu.record\_return('G5678');

END;

/

-- Expected output: Error: User has not borrowed this book or it has already been returned.

-- Test 2: Successful return

BEGIN

-- Set context to user with an active loan

DBMS\_SESSION.SET\_CONTEXT('USER\_CONTEXT', 'USER\_ID', 'USER009');

foundicu.record\_return('H9012');

END;

/

-- Expected output: Book return recorded successfully.

**Complete Package Implementation**

CREATE OR REPLACE PACKAGE foundicu AS

PROCEDURE insert\_loan(p\_signature IN copies.signature%TYPE);

PROCEDURE insert\_reservation(p\_isbn IN editions.isbn%TYPE, p\_date IN DATE);

PROCEDURE record\_return(p\_signature IN copies.signature%TYPE);

END foundicu;

/

CREATE OR REPLACE PACKAGE BODY foundicu AS

-- Insert Loan Procedure

PROCEDURE insert\_loan(p\_signature IN copies.signature%TYPE) IS

v\_user\_id users.user\_id%TYPE;

v\_user\_exists NUMBER;

v\_has\_reservation NUMBER;

v\_town stops.town%TYPE;

v\_province stops.province%TYPE;

v\_stopdate DATE;

v\_copy\_available NUMBER;

v\_loan\_count NUMBER;

v\_is\_banned NUMBER;

v\_max\_loans NUMBER := 5; -- Assuming 5 is the maximum number of loans allowed

BEGIN

-- Get current user ID (assuming it's stored in a context or session)

v\_user\_id := SYS\_CONTEXT('USER\_CONTEXT', 'USER\_ID');

-- Check if user exists

SELECT COUNT(\*) INTO v\_user\_exists

FROM users

WHERE user\_id = v\_user\_id;

IF v\_user\_exists = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Error: User does not exist in the system.');

RETURN;

END IF;

-- Check if there is a reservation for this copy for the current user as of today

SELECT COUNT(\*), MAX(town), MAX(province), MAX(stopdate)

INTO v\_has\_reservation, v\_town, v\_province, v\_stopdate

FROM loans

WHERE signature = p\_signature

AND user\_id = v\_user\_id

AND TRUNC(SYSDATE) = TRUNC(stopdate)

AND return IS NULL

AND type = 'R'; -- Assuming 'R' is the code for reservation

IF v\_has\_reservation > 0 THEN

-- Convert reservation to loan

UPDATE loans

SET type = 'L', -- Assuming 'L' is the code for loan

time = 0 -- Reset time counter

WHERE signature = p\_signature

AND user\_id = v\_user\_id

AND TRUNC(stopdate) = TRUNC(SYSDATE)

AND type = 'R';

DBMS\_OUTPUT.PUT\_LINE('Reservation converted to loan successfully.');

ELSE

-- Check if copy is available for two weeks

SELECT COUNT(\*)

INTO v\_copy\_available

FROM copies

WHERE signature = p\_signature

AND deregistered IS NULL

AND NOT EXISTS (

SELECT 1

FROM loans

WHERE signature = p\_signature

AND return IS NULL

AND TRUNC(stopdate) BETWEEN TRUNC(SYSDATE) AND TRUNC(SYSDATE) + 14

);

IF v\_copy\_available = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Copy is not available for the next two weeks.');

RETURN;

END IF;

-- Check if user has reached loan limit

SELECT COUNT(\*)

INTO v\_loan\_count

FROM loans

WHERE user\_id = v\_user\_id

AND type = 'L'

AND return IS NULL;

IF v\_loan\_count >= v\_max\_loans THEN

DBMS\_OUTPUT.PUT\_LINE('Error: User has reached the maximum number of loans.');

RETURN;

END IF;

-- Check if user is banned

SELECT COUNT(\*)

INTO v\_is\_banned

FROM users

WHERE user\_id = v\_user\_id

AND ban\_up2 IS NOT NULL

AND ban\_up2 >= SYSDATE;

IF v\_is\_banned > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Error: User is currently banned from borrowing.');

RETURN;

END IF;

-- Find current service (town, province, date)

BEGIN

SELECT town, province, taskdate

INTO v\_town, v\_province, v\_stopdate

FROM services

WHERE TRUNC(taskdate) = TRUNC(SYSDATE)

AND ROWNUM = 1;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: No service available today.');

RETURN;

END;

-- Insert new loan

INSERT INTO loans (signature, user\_id, stopdate, town, province, type, time, return)

VALUES (p\_signature, v\_user\_id, v\_stopdate, v\_town, v\_province, 'L', 0, NULL);

DBMS\_OUTPUT.PUT\_LINE('Loan created successfully.');

END IF;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END insert\_loan;

-- Insert Reservation Procedure

PROCEDURE insert\_reservation(p\_isbn IN editions.isbn%TYPE, p\_date IN DATE) IS

v\_user\_id users.user\_id%TYPE;

v\_user\_exists NUMBER;

v\_reservation\_count NUMBER;

v\_is\_banned NUMBER;

v\_max\_reservations NUMBER := 3; -- Assuming 3 is the maximum number of reservations allowed

v\_town stops.town%TYPE;

v\_province stops.province%TYPE;

v\_service\_date DATE;

v\_available\_signature copies.signature%TYPE;

v\_service\_exists NUMBER;

BEGIN

-- Get current user ID (assuming it's stored in a context or session)

v\_user\_id := SYS\_CONTEXT('USER\_CONTEXT', 'USER\_ID');

-- Check if user exists

SELECT COUNT(\*) INTO v\_user\_exists

FROM users

WHERE user\_id = v\_user\_id;

IF v\_user\_exists = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Error: User does not exist in the system.');

RETURN;

END IF;

-- Check if user has reached reservation limit

SELECT COUNT(\*)

INTO v\_reservation\_count

FROM loans

WHERE user\_id = v\_user\_id

AND type = 'R' -- Assuming 'R' is the code for reservation

AND stopdate >= SYSDATE;

IF v\_reservation\_count >= v\_max\_reservations THEN

DBMS\_OUTPUT.PUT\_LINE('Error: User has reached the maximum number of reservations.');

RETURN;

END IF;

-- Check if user is banned

SELECT COUNT(\*)

INTO v\_is\_banned

FROM users

WHERE user\_id = v\_user\_id

AND ban\_up2 IS NOT NULL

AND ban\_up2 >= SYSDATE;

IF v\_is\_banned > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Error: User is currently banned from borrowing.');

RETURN;

END IF;

-- Check if service exists for the specified date

SELECT COUNT(\*)

INTO v\_service\_exists

FROM services

WHERE TRUNC(taskdate) = TRUNC(p\_date);

IF v\_service\_exists = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Error: No service available on the specified date.');

RETURN;

END IF;

-- Find a copy that is available for the next two weeks

BEGIN

SELECT c.signature

INTO v\_available\_signature

FROM copies c

JOIN editions e ON c.isbn = e.isbn

WHERE e.isbn = p\_isbn

AND c.deregistered IS NULL

AND NOT EXISTS (

SELECT 1

FROM loans l

WHERE l.signature = c.signature

AND l.return IS NULL

AND l.stopdate BETWEEN TRUNC(p\_date) AND TRUNC(p\_date) + 14

)

AND ROWNUM = 1;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Error: No available copy of this edition for the specified date range.');

RETURN;

END;

-- Get service information for the requested date

SELECT town, province, taskdate

INTO v\_town, v\_province, v\_service\_date

FROM services

WHERE TRUNC(taskdate) = TRUNC(p\_date)

AND ROWNUM = 1;

-- Insert reservation

INSERT INTO loans (signature, user\_id, stopdate, town, province, type, time, return)

VALUES (v\_available\_signature, v\_user\_id, v\_service\_date, v\_town, v\_province, 'R', 0, NULL);

DBMS\_OUTPUT.PUT\_LINE('Reservation created successfully for copy ' || v\_available\_signature || ' on ' || TO\_CHAR(p\_date, 'DD-MM-YYYY'));

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END insert\_reservation;

-- Record Books Returning Procedure

PROCEDURE record\_return(p\_signature IN copies.signature%TYPE) IS

v\_user\_id users.user\_id%TYPE;

v\_loan\_exists NUMBER;

BEGIN

-- Get current user ID (assuming it's stored in a context or session)

v\_user\_id := SYS\_CONTEXT('USER\_CONTEXT', 'USER\_ID');

-- Check if user has borrowed this book

SELECT COUNT(\*)

INTO v\_loan\_exists

FROM loans

WHERE signature = p\_signature

AND user\_id = v\_user\_id

AND type = 'L' -- Assuming 'L' is the code for loan

AND return IS NULL;

IF v\_loan\_exists > 0 THEN

-- Record return

UPDATE loans

SET return = SYSDATE

WHERE signature = p\_signature

AND user\_id = v\_user\_id

AND type = 'L'

AND return IS NULL;

DBMS\_OUTPUT.PUT\_LINE('Book return recorded successfully.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Error: User has not borrowed this book or it has already been returned.');

END IF;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END record\_return;

END foundicu;

/