AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

408/1, Kuratoli, Khilkhet, Dhaka 1229,
Bangladesh

Project Title: Plan My Trip

Course Title: Advance Database Management System

Course Code: CSC 4181 Section: B

Semester: Spring 2023 - 2024 Course Teacher: **REZWAN AHMED**

Group Name/No.: **A**

No	Name	ID	Program	Signature
1	**Fatin Noor**	**21-44729-1**	BSc [CSE]	
2	**Iftekhar Uddin Mullick**	**21-44649-1**	BSc [CSE]	
3	**Md. Shahriar Mahmud**	**21-44498-1**	BSc [CSE]	
4	**Pranta Hossen**	**21-44472-1**	BSc [CSE]	

Faculty use only		
FACULTYCOMMENTS	**Marks Obtained**	
	Total Marks	

1. Project Description

The name of the project that was developed for this course is 'PlanMyTrip', a website that will provide Hotels and Travel Providers to provide a complete solution to the users to purchase travel plans from providers across the globe.

The scope of the project has been moderated to ensure that all the objective written in the Final Deliverables are delivered in a fruitful manner. To accommodate the requirements properly within the provided time. Only the following two users have been developed properly:

- Admin
- User

The features are in the project are described based on the roles one by one:

Admin

- Change password
- View users
- Search users
- Delete users
- View users balance
- View user balance asc and desc order
- Create hotel mgt
- Add tour packages
- Delete tour packages
- Search tour packages
- Add contact info
- Search contact info

User

- Profile
- Edit profile
- Deposit money
- See balance
- View transection
- View contact info
- Search contact info
- View tour packages
- Search tour packages
- See transection

2. **List of PLSQL Objects**

The following website has the following PLSQL objects prepared:

PLSQL View

i. View users Table
ii. View balance from wallet Table
iii. View contactinfo Table
iv. View tourpackages Table
v. View review Table
vi. View transaction Table
vii. View Details of 3 Users who has the highest wallet balance
viii. View Details of 3 Users who has the lowest wallet balance
ix. Criteria Based Search View on Review

PLSQL Function

i. Check Current Password of the user who is logged in
ii. Update Current Password of any user who is logged in
iii. Search if the hotel exists or not
iv. Insert New Hotel Data
v. Insert New Hotel Authentication Data
vi. Remove User Data
vii. Remove User Authentication Data
viii. Remove User Wallet Data

PLSQL Procedure

i. Update Password of Logged in User by using Function (i) and (ii)
ii. Insert Contact Information
iii. Insert Hotel Package information after checking if the hotel exists or not using Function (iii)
iv. Create Hotel account using the Function (iv) and (v)
v. Remove User information by using Function (vi), (vii) and (viii)
vi. Delete Tour Packages
vii. Search Users
viii. Create User Accounts
ix. Search Contact Information
x. Search Tour Packages using Destination
xi. Check Current Wallet Balance of the user who is logged in
xii. Update Wallet Balance of the user who is logged in
xiii. Update User Profile Data
xiv. Add Reviews

PLSQL Sequence

i. Generate HotelId for Hotel Account Creation
ii. Generate ReviewId for Review Insertion
iii. Generate LogId for Log Insertion of User Table Operations

PLSQL Trigger

i. Create new HotelId using sequence (i)

ii. Create new ReviewId using sequence (ii)
iii. Create Logs of Insertion, Update and Delete on User Table using sequence (iii)
iv. Prevent Wallet Balance Update when the value exceeds 10,00,000
v. Prevent Wallet Balance Update when the time is between 1 AM to 7 AM

PLSQL Locking

i. Lock User Reviews if user has provided 3 reviews in a day

PLSQL Packages

- Package of Tour Package Management

3. Code of PLSQL Objects

To display the data inside the database, PLSQL View objects where generated in the database and those PLSQL Views were accessed using PLSQL Procedures. The SQL of the PLSQL View and relevant PLSQL Procedure were used are provided down below:

- View 1: View users Table

PLSQL View

```
CREATE OR REPLACE VIEW vw_users AS
SELECT *
FROM users;
```

PLSQL Procedure

```
CREATE OR REPLACE PROCEDURE fetch_users (result OUT SYS_REFCURSOR)
AS
BEGIN
      OPEN result FOR
      SELECT username, name, email, phonenumber, dob, profilepic
      FROM vw_users;
END;
```

- View 2: View balance from wallet Table

PLSQL View

```
CREATE OR REPLACE VIEW vw_wallet AS
SELECT *
FROM wallet;
```

PLSQL Procedure

```
CREATE OR REPLACE PROCEDURE fetch_users_wallet_info (result OUT
SYS_REFCURSOR) AS
BEGIN
      OPEN result FOR
      SELECT username, phonenumber, balance, paymentmethod
      FROM vw_wallet;
END;
```

- View 3: View contactinfo Table

PLSQL View

```
CREATE OR REPLACE VIEW vw_contactinfo AS
SELECT *
FROM contactinfo;
```

PLSQL Procedure

```
CREATE OR REPLACE PROCEDURE fetch_contact_info (result OUT
SYS_REFCURSOR) AS
BEGIN
    OPEN result FOR
    SELECT contactname, email, phonenumber
    FROM vw_contactinfo;
END;
```

- View 4: View tourpackages Table

PLSQL View

```
CREATE OR REPLACE VIEW vw_tourpackages AS
SELECT *
FROM tourpackages;
```

PLSQL Procedure

```
CREATE OR REPLACE PROCEDURE fetch_tour_packages (result OUT
SYS_REFCURSOR) AS
BEGIN
    OPEN result FOR
    SELECT packageid, hotelid, packagename, destination, days,
      meals, price
    FROM vw_tourpackages;
END;
```

- View 5: View review Table

PLSQL View

```
CREATE OR REPLACE VIEW vw_review AS
SELECT *
FROM review;
```

PLSQL Procedure

```
CREATE OR REPLACE PROCEDURE fetch_review (result OUT SYS_REFCURSOR)
AS
BEGIN
    OPEN result FOR
    SELECT id, username, rating, description, servicetype,
reviewdate
    FROM vw_review;
END;
```

- View 6: View transaction Table

PLSQL View

```
CREATE OR REPLACE VIEW vw_transaction AS
SELECT *
FROM transaction;
```

PLSQL Procedure

```
CREATE OR REPLACE PROCEDURE fetch_transaction (result OUT
SYS_REFCURSOR) AS
BEGIN
    OPEN result FOR
    SELECT paymentid, username, usermethod, cardnumber,
mobilenumber, amount
    FROM vw_transaction;
END;
```

- View 7: View Details of 3 Users who has the highest wallet balance

PLSQL View

```
CREATE OR REPLACE VIEW vw_max_three_balance AS
SELECT users.name, users.email, users.phonenumber, wallet.balance
FROM users, wallet
WHERE users.username = wallet.username AND users.username
IN
      (SELECT username
      FROM (
            SELECT username
            FROM wallet
            ORDER BY balance DESC)
      WHERE ROWNUM <= 3)
```

PLSQL Procedure

```
CREATE OR REPLACE PROCEDURE fetch_max_three_balance (result OUT
SYS_REFCURSOR) AS
BEGIN
    OPEN result FOR
    SELECT name, email, phonenumber, balance
    FROM vw_max_three_balance;
END;
```

- View 8: View Details of 3 Users who has the lowest wallet balance

PLSQL View

```
CREATE OR REPLACE VIEW vw_min_three_balance AS
SELECT users.name, users.email, users.phonenumber, wallet.balance
FROM users, wallet WHERE users.username = wallet.username AND
users.username IN
      (SELECT username
      FROM
            (SELECT username
            FROM wallet
            ORDER BY balance ASC)
      WHERE ROWNUM <= 3)
```

PLSQL Procedure

```
CREATE OR REPLACE PROCEDURE fetch_min_three_balance (result OUT
SYS_REFCURSOR) AS
BEGIN
    OPEN result FOR
    SELECT name, email, phonenumber, balance
    FROM vw_min_three_balance;
END;
```

- View 9: Criteria Based Search View on Review

PLSQL Procedure

```
CREATE OR REPLACE PROCEDURE fetch_filtered_reviews(
    p_rating      IN review.rating%TYPE,
    p_servicetype IN review.servicetype%TYPE,
    p_start_date  IN review.reviewdate%TYPE,
    p_end_date    IN review.reviewdate%TYPE,
    p_cursor      OUT SYS_REFCURSOR
) IS
BEGIN
    OPEN p_cursor FOR
        SELECT id, username, rating, description, servicetype,
reviewdate
        FROM review
        WHERE (rating = p_rating)
          AND (servicetype = p_servicetype)
          AND (reviewdate >= p_start_date)
          AND (reviewdate <= p_end_date);
END;
```

For ease of understanding the relevance of each PLSQL Objects (Procedure, Function, Trigger etc) the codes have been clustered into operations. The PLSQL codes are provided down below:

- **Operation 1:** Update Password of Admin

  PLSQL Function (i) - Check Current Password of the user who is logged in

```
CREATE OR REPLACE FUNCTION VerifyCurrentPassword(uname
authinfo.username%type, pass authinfo.password%type)
RETURN BOOLEAN
IS
  current_password VARCHAR2(255);
BEGIN
  SELECT password INTO current_password FROM authinfo WHERE username
= uname;

  IF current_password = pass THEN
    RETURN TRUE;
  ELSE
    RETURN FALSE;
  END IF;

EXCEPTION
  WHEN NO_DATA_FOUND OR OTHERS THEN
    RETURN FALSE;
END;
```

  PLSQL Function (ii) - Update Current Password of the user logged in

```
CREATE OR REPLACE FUNCTION UpdatePassword(uname
authinfo.username%type, new_pass VARCHAR2)
RETURN BOOLEAN
IS
BEGIN
  UPDATE authinfo SET password = new_pass WHERE username = uname;
  RETURN TRUE;

EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
    RETURN FALSE;
END;
```

PLSQL Procedure (i) - Update Password of Logged in User by using Function (i) and (ii)

```
CREATE OR REPLACE PROCEDURE ChangePassword(uname
authinfo.username%type, pass authinfo.password%type, new_pass VARCHAR2,
result_out OUT NUMBER)
IS
  current_pass_verified BOOLEAN;
BEGIN
  current_pass_verified := VerifyCurrentPassword(uname, pass);

  IF current_pass_verified THEN
    IF UpdatePassword(uname, new_pass) THEN
      DBMS_OUTPUT.PUT_LINE('Password updated successfully.');
      result_out := 0;
    ELSE
      DBMS_OUTPUT.PUT_LINE('Failed to update password.');
      result_out := 1;
    END IF;
  ELSE
    DBMS_OUTPUT.PUT_LINE('Current password is incorrect.');
    result_out := 2;
  END IF;

EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error occurred: ' || SQLERRM);
    result_out := 3;
END;
```

- **Operation 2:** Add Contact Information of Essential Personals

PLSQL Procedure (ii) - Insert Contact Information

```
CREATE OR REPLACE PROCEDURE AddContact(
      cname contactinfo.contactname%type,
      cemail contactinfo.email%type,
      cphone contactinfo.phonenumber%type,
      result_out OUT NUMBER) IS
BEGIN
  INSERT INTO contactinfo(contactname, email, phonenumber)
  VALUES (cname, cemail, cphone);
  result_out := 1;

EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
    result_out := 0;
END;
```

- **Operation 3:** Add Hotel Packages of Registered Hotels

PLSQL Function (iii) - Search if the hotel exists or not

```
CREATE OR REPLACE FUNCTION CheckHotelID(
      hotel_id hotelmgt.hotelid%type)
RETURN BOOLEAN IS
  v_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_count FROM hotelmgt WHERE hotelid =
hotel_id;

  IF v_count > 0 THEN
    RETURN TRUE;
  ELSE
    RETURN FALSE;
  END IF;

EXCEPTION
  WHEN OTHERS THEN
    RETURN FALSE;
END;
```

PLSQL Procedure (iii) - Insert Hotel Package information after checking if the hotel exists or not using Function (iii)

```
CREATE OR REPLACE PROCEDURE AddTourPackage(
      pid tourpackages.packageid%type,
      hid tourpackages.hotelid%type,
      pname tourpackages.packagename%type,
      dest tourpackages.destination%type,
      num_days tourpackages.days%type,
      meal_plan tourpackages.meals%type,
      pack_price tourpackages.price%type,
      result_out OUT NUMBER)
IS
BEGIN
  IF CheckHotelID(hid) THEN
    INSERT INTO tourpackages(packageid, hotelid, packagename,
destination, days, meals, price)
    VALUES (pid, hid, pname, dest, num_days, meal_plan, pack_price);

    result_out := 1;
  ELSE
    result_out := 0;
  END IF;
```

```
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
    result_out := -1;
END;
```

- **Operation 4:** Create Hotel Accounts on the system

PLSQL Sequence (i) - Generate HotelId for Hotel Account Creation

```
CREATE SEQUENCE hotelid_seq
START WITH 1
INCREMENT BY 1
NOCACHE;
```

PLSQL Trigger (i) - Create new HotelId using sequence (ii)

```
CREATE OR REPLACE TRIGGER hotelid_bfr_insert
BEFORE INSERT ON hotelmgt
FOR EACH ROW
BEGIN
  IF :new.hotelid IS NULL THEN
    SELECT hotelid_seq.NEXTVAL INTO :new.hotelid FROM dual;
  END IF;
END;
```

PLSQL Function (iv) - Insert New Hotel Data

```
CREATE OR REPLACE FUNCTION InsertHotelMgt(
    u_username hotelmgt.username%type,
    u_email hotelmgt.email%type,
    u_hotelname hotelmgt.hotelname%type,
    u_phone hotelmgt.phonenumber%type,
    u_logo hotelmgt.logo%type)
RETURN BOOLEAN IS
BEGIN
    INSERT INTO hotelmgt(username, email, hotelname, phonenumber,
logo)
    VALUES (u_username, u_email, u_hotelname, u_phone, u_logo);
    RETURN TRUE;

EXCEPTION
    WHEN OTHERS THEN
        RETURN FALSE;
END;
```

PLSQL Function (v) - Insert New Hotel Authentication Data

```
CREATE OR REPLACE FUNCTION InsertAuthInfo(
    u_username authinfo.username%type,
    u_password authinfo.password%type,
    u_role authinfo.role%type)
RETURN BOOLEAN IS
BEGIN
    INSERT INTO authinfo(username, password, role)
    VALUES (u_username, u_password, u_role);
    RETURN TRUE;

EXCEPTION
    WHEN OTHERS THEN
        RETURN FALSE;
END;
```

PLSQL Procedure (iv) - Create Hotel account using the Function (iv) and (v)

```
CREATE OR REPLACE PROCEDURE AddHotel(
    uname hotelmgt.username%type,
    uemail hotelmgt.email%type,
    hname hotelmgt.hotelname%type,
    phone hotelmgt.phonenumber%type,
    hlogo hotelmgt.logo%type,
    auth_username authinfo.username%type,
    auth_password authinfo.password%type,
    auth_role authinfo.role%type,
    result_out OUT NUMBER
) IS
    hotel_result BOOLEAN;
    auth_result BOOLEAN;
BEGIN
    hotel_result := InsertHotelMgt(uname, uemail, hname, phone, hlogo);
    auth_result := InsertAuthInfo(auth_username, auth_password,
auth_role);

    IF hotel_result AND auth_result THEN
        result_out := 0;
    ELSE
        ROLLBACK;
        result_out := 1;
    END IF;

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        result_out := 2;
END;
```

- **Operation 5:** Remove Users from the System by Admin

PLSQL Function (vi) - Remove User Data

```
CREATE OR REPLACE FUNCTION DeleteFromUsers(
      u_username users.username%type)
RETURN BOOLEAN IS
BEGIN
    DELETE FROM users WHERE username = u_username;

    SELECT COUNT(*) INTO count_exists FROM users WHERE username =
u_username;

    IF count_exists = 0 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;

EXCEPTION
    WHEN OTHERS THEN
        RETURN FALSE;
END;
```

PLSQL Function (vii) - Remove User Authentication Data

```
CREATE OR REPLACE FUNCTION DeleteFromAuthInfo(
      u_username authinfo.username%type)
RETURN BOOLEAN IS
BEGIN
    DELETE FROM authinfo WHERE username = u_username;

    SELECT COUNT(*) INTO count_exists FROM authinfo WHERE username =
u_username;

    IF count_exists = 0 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;

EXCEPTION
    WHEN OTHERS THEN
        RETURN FALSE;
END;
```

PLSQL Function (viii) - Remove User Wallet Data

```
CREATE OR REPLACE FUNCTION DeleteFromWallet(
      u_username VARCHAR2)
RETURN BOOLEAN IS
BEGIN
    DELETE FROM wallet WHERE username = u_username;

    SELECT COUNT(*) INTO count_exists FROM wallet WHERE username =
u_username;

    IF count_exists = 0 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;

EXCEPTION
    WHEN OTHERS THEN
        RETURN FALSE;
END;
```

PLSQL Procedure (v) - Remove User information by using Function (vi), (vii) and (viii)

```
CREATE OR REPLACE PROCEDURE DeleteUserAccount(
    p_username IN users.username%TYPE,
    result_out OUT NUMBER
)
IS
    result_wallet BOOLEAN;
    result_authinfo BOOLEAN;
    result_users BOOLEAN;
BEGIN
    result_wallet := DeleteFromWallet(p_username);
    result_authinfo := DeleteFromAuthInfo(p_username);
    result_users := DeleteFromUsers(p_username);

    IF result_wallet AND result_authinfo AND result_users THEN
        result_out := 1;
    ELSE
        ROLLBACK;
        result_out := 0;
    END IF;

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        result_out := -1;
END;
```

- **Operation 6:** Remove Tour Packages of Hotels by Admin

  PLSQL Procedure (vi) - Delete Tour Packages

```
CREATE OR REPLACE PROCEDURE DeleteTourPackage(
      p_packageid tourpackages.packageid%type,
      result_out OUT NUMBER)
IS
    count_exists NUMBER;
BEGIN
    DELETE FROM tourpackages WHERE packageid = p_packageid;
    SELECT COUNT(*) INTO count_exists FROM tourpackages WHERE packageid
= p_packageid;

    IF count_exists = 0 THEN
        result_out := 1;
    ELSE
        ROLLBACK;
        result_out := 0;
    END IF;

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        result_out := -1;
END;
```

- **Operation 7:** Search Users in the System

  PLSQL Procedure (vii) - Search Users

```
CREATE OR REPLACE PROCEDURE SearchUserByUsername(
    p_username IN users.username%TYPE,
    p_user_cursor OUT SYS_REFCURSOR,
    result_out OUT NUMBER
) IS
BEGIN
    OPEN p_user_cursor FOR
    SELECT username, name, email, phonenumber, dob, profilepic
    FROM users
    WHERE username = p_username;

    IF p_user_cursor%ISOPEN THEN
        result_out := 1;
    ELSE
        result_out := 0;
    END IF;

EXCEPTION
    WHEN OTHERS THEN
        result_out := -1;
END;
```

- **Operation 8:** User Signup in the System

PLSQL Procedure (viii) - Create User Accounts

```
CREATE OR REPLACE PROCEDURE OpenUserAccount(
    p_username IN users.username%TYPE,
    p_name IN users.name%TYPE,
    p_email IN users.email%TYPE,
    p_phonenumber IN users.phonenumber%TYPE,
    p_dob IN users.dob%TYPE,
    p_profilepic IN users.profilepic%TYPE,
    p_password IN authinfo.password%TYPE,
    p_role IN authinfo.role%TYPE,
    p_wallet_phonenumber IN wallet.phonenumber%TYPE,
    result_out OUT NUMBER
)
IS
BEGIN
    INSERT INTO users(username, name, email, phonenumber, dob,
profilepic)
    VALUES (p_username, p_name, p_email, p_phonenumber, p_dob,
p_profilepic);
    INSERT INTO authinfo(username, password, role)
    VALUES (p_username, p_password, p_role);
    INSERT INTO wallet(username, phonenumber, balance,
paymentmethod)
    VALUES (p_username, p_wallet_phonenumber, 0, 'initial');
    result_out := 1;

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        result_out := 0;
END;
```

- **Operation 9:** Search Contact Information in the System

  PLSQL Procedure (ix) - Search Contact Information

```
CREATE OR REPLACE PROCEDURE SearchContactByName(
    p_contactname IN contactinfo.contactname%TYPE,
    p_contact_cursor OUT SYS_REFCURSOR,
    result_out OUT NUMBER
) IS
BEGIN
    OPEN p_contact_cursor FOR
    SELECT contactname, email, phonenumber
    FROM contactinfo
    WHERE contactname LIKE p_contactname || '%';

    IF p_contact_cursor%ISOPEN THEN
        result_out := 1;
    ELSE
        result_out := 0;
    END IF;

EXCEPTION
    WHEN OTHERS THEN
        result_out := -1;
END;
```

- **Operation 10:** Search Tour Packages offered on the System
  PLSQL Procedure (x) - Search Tour Packages using Destination

```
CREATE OR REPLACE PROCEDURE SearchTourPackagesByDestination(
    p_destination IN tourpackages.destination%TYPE,
    p_tour_cursor OUT SYS_REFCURSOR,
    result_out OUT NUMBER
) IS
BEGIN
    OPEN p_tour_cursor FOR
    SELECT packageid, hotelid, packagename, destination, days, meals,
price
    FROM tourpackages
    WHERE destination LIKE p_destination || '%';

    IF p_tour_cursor%ISOPEN THEN
        result_out := 1;
    ELSE
        result_out := 0;
    END IF;

EXCEPTION
    WHEN OTHERS THEN
        result_out := -1;
END;
```

- **Operation 11:** Add Wallet Balance by Users

PLSQL Procedure (xi) - Check Current Wallet Balance of the user who is logged in

```
CREATE OR REPLACE PROCEDURE GetOldBalance (
    p_phonenumber IN wallet.phonenumber%type,
    p_balance OUT wallet.balance%type,
    result_out OUT NUMBER
) AS
BEGIN
    SELECT balance INTO p_balance
    FROM wallet
    WHERE phonenumber = p_phonenumber;

    result_out := 1;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        raise_application_error(-20001, 'No user found with the
provided phone number.');
    WHEN OTHERS THEN
        raise_application_error(-20002, 'An error occurred while
fetching old balance: ' || SQLERRM);
END;
```

PLSQL Procedure (xii) - Update Wallet Balance of the user who is logged in

```
CREATE OR REPLACE PROCEDURE update_balance_by_phonenumber (
    p_phonenumber IN wallet.phonenumber%type,
    p_balance IN wallet.balance%type,
    p_paymentmethod IN wallet.paymentmethod%type
) AS
BEGIN
    UPDATE wallet
    SET balance = p_balance, paymentmethod = p_paymentmethod
    WHERE phonenumber = p_phonenumber;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        raise_application_error(-20001, 'No user found with the
provided phone number.');
    WHEN OTHERS THEN
        raise_application_error(-20002, 'An error occurred while
updating balance: ' || SQLERRM);
END;
```

PLSQL Trigger (iv) - Prevent Wallet Balance Update when the value exceeds 10,00,000

```
CREATE OR REPLACE TRIGGER PreventExcessiveBalance
BEFORE UPDATE ON wallet
FOR EACH ROW
BEGIN
    IF :NEW.balance > 1000000 THEN
        raise_application_error(-20002, 'Balance cannot exceed
10,00,000!');
    END IF;
END;
```

PLSQL Trigger (v) - Prevent Wallet Balance Update when the time is between 1 AM to 7 AM

```
CREATE OR REPLACE TRIGGER PreventWalletUpdate
BEFORE UPDATE ON wallet
DECLARE
    v_current_hour NUMBER;
BEGIN
    SELECT TO_NUMBER(TO_CHAR(SYSDATE, 'HH24')) INTO v_current_hour
FROM dual;

    IF v_current_hour >= 1 AND v_current_hour <= 7 THEN
        raise_application_error(-20001, 'Wallet update is not
allowed between 01:00 AM and 07:00 AM.');
    END IF;
END;
```

- **Operation 12:** Update User Profile Information

  PLSQL Procedure (xiii) - Update User Profile Data

```
CREATE OR REPLACE PROCEDURE update_user_info (
    p_username IN users.username%TYPE,
    p_name IN users.name%TYPE,
    p_email IN users.email%TYPE,
    p_dob IN users.dob%TYPE
) AS
BEGIN
    UPDATE users
    SET name = p_name,
        email = p_email,
        dob = p_dob
    WHERE username = p_username;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        raise_application_error(-20001, 'No user found with the
provided username.');
    WHEN OTHERS THEN
        raise_application_error(-20002, 'An error occurred while
updating user information: ' || SQLERRM);
END;
```

- **Operation 13:** Add User Review of Products

  PLSQL Sequence (ii) - Generate ReviewId for Review Insertion

```
CREATE SEQUENCE ReviewIdSeq
START WITH 1
INCREMENT BY 1
NOCACHE;
```

  PLSQL Trigger (ii) - Create new ReviewId using sequence (ii)

```
CREATE OR REPLACE TRIGGER ReviewIdBfrInsert
BEFORE INSERT ON review
FOR EACH ROW
BEGIN
  IF :new.id IS NULL THEN
    SELECT ReviewIdSeq.NEXTVAL INTO :new.id FROM dual;
  END IF;
END;
```

PLSQL Procedure (xiv) - Add Reviews

```
CREATE OR REPLACE PROCEDURE InsertReview (
    p_username IN review.username%TYPE,
    p_rating IN review.rating%TYPE,
    p_description IN review.description%TYPE,
    p_servicetype IN review.servicetype%TYPE
) AS
BEGIN
    INSERT INTO review (id, username, rating, description,
servicetype, reviewdate)
    VALUES (ReviewIdSeq.NEXTVAL, p_username, p_rating,
p_description, p_servicetype, SYSDATE);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        raise_application_error(-20001, 'No user found with the
provided username.');
    WHEN OTHERS THEN
        raise_application_error(-20002, 'An error occurred while
inserting the review: ' || SQLERRM);
END;
```

PLSQL Locking (i) - Lock User Reviews if user has provided 3 reviews in a day

```
CREATE OR REPLACE PROCEDURE InsertReview (
    p_username IN review.username%TYPE,
    p_rating IN review.rating%TYPE,
    p_description IN review.description%TYPE,
    p_servicetype IN review.servicetype%TYPE
) AS
    v_user_reviews_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_user_reviews_count
    FROM review
    WHERE username = p_username
    AND TRUNC(reviewdate) = TRUNC(SYSDATE);

    IF v_user_reviews_count >= 3 THEN
        RAISE_APPLICATION_ERROR(-20001, 'You have already provided 3
reviews today.');
    END IF;

    INSERT INTO review (id, username, rating, description,
servicetype, reviewdate)
    VALUES (ReviewIdSeq.NEXTVAL, p_username, p_rating, p_description,
p_servicetype, SYSDATE);
```

```
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20002, 'No user found with the
provided username.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20003, 'An error occurred while
inserting the review: ' || SQLERRM);
END;
```

- **Operation 14:** System Logging of operations on Users Table

  PLSQL Sequence (iii) - Generate LogId for Log Insertion of User Table Operations

```
CREATE SEQUENCE logid_seq
START WITH 1
INCREMENT BY 1
NOCACHE;
```

  PLSQL Trigger (iii) - Create Logs of Insertion, Update and Delete on User Table using sequence (iii)

```
CREATE OR REPLACE TRIGGER log_trigger
AFTER INSERT OR UPDATE OR DELETE ON users
FOR EACH ROW
BEGIN
    DECLARE
        v_logdata VARCHAR2(255);
    BEGIN
        IF INSERTING THEN
            v_logdata := 'Insertion Completed';
        ELSIF UPDATING THEN
            v_logdata := 'Update Completed';
        ELSIF DELETING THEN
            v_logdata := 'Deletion Completed';
        END IF;

        INSERT INTO log_table (logid, logdata, logdate, logtime)
        VALUES (logid_seq.NEXTVAL, v_logdata, SYSDATE,
CURRENT_TIMESTAMP);
    END;
END;
```

- **Package:**

In our project a package was developed which focused on the PLSQL Objects related to Tour Package Management. Here are the Package Specification and Package Body:

Package Specification

```
CREATE OR REPLACE PACKAGE TourPackageManagement AS
FUNCTION CheckHotelID(
      hotel_id IN hotelmgt.hotelid%TYPE)
RETURN BOOLEAN;

PROCEDURE AddTourPackage(
      pid IN tourpackages.packageid%TYPE,
      hid IN tourpackages.hotelid%TYPE,
      pname IN tourpackages.packagename%TYPE,
      dest IN tourpackages.destination%TYPE,
      num_days IN tourpackages.days%TYPE,
      meal_plan IN tourpackages.meals%TYPE,
      pack_price IN tourpackages.price%TYPE,
      result_out OUT NUMBER);

PROCEDURE DeleteTourPackage(
      p_packageid IN tourpackages.packageid%TYPE,
      result_out OUT NUMBER);

PROCEDURE SearchTourPackagesByDestination(
      p_destination IN tourpackages.destination%TYPE,
      p_tour_cursor OUT SYS_REFCURSOR,
      result_out OUT NUMBER);
END;
```

Package Body

```
CREATE OR REPLACE PACKAGE BODY TourPackageManagement AS

FUNCTION CheckHotelID(hotel_id IN hotelmgt.hotelid%TYPE)
RETURN BOOLEAN
IS
      v_count NUMBER;
BEGIN
      SELECT COUNT(*) INTO v_count
      FROM hotelmgt
      WHERE hotelid = hotel_id;

RETURN v_count > 0;

EXCEPTION
      WHEN OTHERS THEN
```

```
            RETURN FALSE;
END CheckHotelID;

PROCEDURE AddTourPackage(
      pid IN tourpackages.packageid%TYPE,
      hid IN tourpackages.hotelid%TYPE,
      pname IN tourpackages.packagename%TYPE,
      dest IN tourpackages.destination%TYPE,
      num_days IN tourpackages.days%TYPE,
      meal_plan IN tourpackages.meals%TYPE,
      pack_price IN tourpackages.price%TYPE,
      result_out OUT NUMBER)
IS
BEGIN
      IF CheckHotelID(hid) THEN
            INSERT INTO tourpackages(packageid, hotelid, packagename,
destination, days, meals, price)
            VALUES (pid, hid, pname, dest, num_days, meal_plan,
pack_price);
            result_out := 1;
      ELSE
            result_out := 0;
      END IF;
EXCEPTION
      WHEN OTHERS THEN
            ROLLBACK;
            result_out := -1;
END AddTourPackage;

PROCEDURE DeleteTourPackage(
      p_packageid IN tourpackages.packageid%TYPE,
      result_out OUT NUMBER)
IS
      count_exists NUMBER;
BEGIN
      DELETE FROM tourpackages WHERE packageid = p_packageid;
      SELECT COUNT(*) INTO count_exists FROM tourpackages WHERE
packageid = p_packageid;
      IF count_exists = 0 THEN
            result_out := 1;
      ELSE
            ROLLBACK;
            result_out := 0;
      END IF;
EXCEPTION
      WHEN OTHERS THEN
            ROLLBACK;
            result_out := -1;
```

```
END DeleteTourPackage;

PROCEDURE SearchTourPackagesByDestination(
      p_destination IN tourpackages.destination%TYPE,
      p_tour_cursor OUT SYS_REFCURSOR,
      result_out OUT NUMBER)
IS
BEGIN
      OPEN p_tour_cursor FOR
      SELECT packageid, hotelid, packagename, destination, days,
meals, price
      FROM tourpackages
      WHERE destination LIKE p_destination || '%';
      IF p_tour_cursor%ISOPEN THEN
            result_out := 1;
      ELSE
            result_out := 0;
      END IF;
EXCEPTION
      WHEN OTHERS THEN
            result_out := -1;
END SearchTourPackagesByDestination;
END TourPackageManagement;
```

Sample Anonymous Block to Execute Package

```
DECLARE
      result_add NUMBER;
      result_delete NUMBER;
      result_search NUMBER;
      tour_cursor SYS_REFCURSOR;

BEGIN
      TourPackageManagement.AddTourPackage(v_pid, v_hid, v_pname,
v_dest, v_num_days, v_meal_plan, v_pack_price, result_add);

      TourPackageManagement.DeleteTourPackage(v_packageid_to_delete,
result_delete);

      TourPackageManagement.SearchTourPackagesByDestination(v_search
_dest, tour_cursor, result_search);
      LOOP
            FETCH tour_cursor INTO v_pid, v_hid, v_pname, v_dest,
v_num_days, v_meal_plan, v_pack_price;
      EXIT WHEN tour_cursor%NOTFOUND;
      END LOOP;
      CLOSE tour_cursor;
```

```
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
```