DBMS PROJECT

```
CREATE TABLE Users (
   user_id NUMBER PRIMARY KEY,
   username VARCHAR2(50) UNIQUE NOT NULL,
   email VARCHAR2(100) UNIQUE NOT NULL,
   created_at DATE DEFAULT SYSDATE NOT NULL
);

CREATE TABLE Budgets (
   budget_id NUMBER PRIMARY KEY,
   user_id NUMBER NOT NULL,
   budget_name VARCHAR2(100),
   total_budget DECIMAL(10, 2),
   start_date DATE,
   end_date DATE,
   CONSTRAINT fk_budgets_users FOREIGN KEY (user_id) REFERENCES Users(user_id)
);

CREATE TABLE Categories (
   category_id NUMBER PRIMARY KEY,
   category_name VARCHAR2(50) NOT NULL
);

CREATE TABLE Expenses (
   expense_id NUMBER PRIMARY KEY,
   budget_id NUMBER NOT NULL,
   category_id NUMBER NOT NULL,
   amount DECIMAL(10, 2),
   expense_date DATE DEFAULT SYSDATE NOT NULL,
   description VARCHAR2(255),
   CONSTRAINT fk_expenses_budgets FOREIGN KEY (budget_id) REFERENCES
Budgets(budget_id),
   CONSTRAINT fk_expenses_categories FOREIGN KEY (category_id) REFERENCES
Categories(category_id)
);
```

ADD A NEW USER

```
CREATE OR REPLACE PROCEDURE Add_User (
   p_username IN Users.username%TYPE,
   p_email IN Users.email%TYPE
)
AS
   l_user_id Users.user_id%TYPE;
BEGIN
   SELECT user_id_seq.NEXTVAL INTO l_user_id FROM DUAL;
   INSERT INTO Users (user_id, username, email)
   VALUES (l_user_id, p_username, p_email);
   COMMIT;
   DBMS_OUTPUT.PUT_LINE('New user added with ID: ' || l_user_id);
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
```

CREATE A NEW BUDGET

```
CREATE OR REPLACE PROCEDURE Create_Budget (
   p_user_id IN Budgets.user_id%TYPE,
   p_budget_name IN Budgets.budget_name%TYPE,
   p_total_budget IN Budgets.total_budget%TYPE,
   p_start_date IN Budgets.start_date%TYPE,
   p_end_date IN Budgets.end_date%TYPE
)
AS
   l_budget_id Budgets.budget_id%TYPE;
BEGIN
   SELECT budget_id_seq.NEXTVAL INTO l_budget_id FROM DUAL;
   INSERT INTO Budgets (budget_id, user_id, budget_name, total_budget, start_date,
end_date)
   VALUES (l_budget_id, p_user_id, p_budget_name, p_total_budget, p_start_date,
p_end_date);
   COMMIT;
   DBMS_OUTPUT.PUT_LINE('Budget created successfully with ID: ' || l_budget_id);
EXCEPTION
   WHEN OTHERS THEN
```

```
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
```

RECORD AN EXPENSE

```
CREATE OR REPLACE PROCEDURE Record_Expense (
  p_budget_id IN Expenses.budget_id%TYPE,
  p_category_id IN Expenses.category_id%TYPE,
  p_amount IN Expenses.amount%TYPE,
  p_description IN Expenses.description%TYPE
)
AS
BEGIN
  INSERT INTO Expenses (expense_id, budget_id, category_id, amount, description)
  VALUES (expense_id_seq.NEXTVAL, p_budget_id, p_category_id, p_amount,
p_description);
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Expense recorded successfully.');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
```

CALCULATE TOTAL EXPENSES

```
CREATE OR REPLACE FUNCTION Total_Expenses (
  p_budget_id IN Expenses.budget_id%TYPE
) RETURN DECIMAL
AS
  l_total DECIMAL(10, 2) := 0;
BEGIN
  SELECT SUM(amount) INTO l_total FROM Expenses WHERE budget_id = p_budget_id;
  RETURN l_total;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN 0;
  WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20001, 'An error occurred calculating total expenses: '
|| SQLERRM);
END;
```

Cursor for displaying expenses

```
CREATE OR REPLACE PROCEDURE List_Expenses_By_Budget (
   p_budget_id IN Expenses.budget_id%TYPE
) AS
   CURSOR c_expenses IS
      SELECT e.amount, e.description, e.expense_date, c.category_name
      FROM Expenses e
      JOIN Categories c ON e.category_id = c.category_id
      WHERE e.budget_id = p_budget_id
      ORDER BY e.expense_date DESC;

   l_amount Expenses.amount%TYPE;
   l_description Expenses.description%TYPE;
   l_expense_date Expenses.expense_date%TYPE;
   l_category_name Categories.category_name%TYPE;
BEGIN
   OPEN c_expenses;
   LOOP
      FETCH c_expenses INTO l_amount, l_description, l_expense_date,
l_category_name;
      EXIT WHEN c_expenses%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Amount: ' || l_amount || ' Description: ' ||
l_description ||
                           ' Date: ' || l_expense_date || ' Category: ' || l_category_name);
   END LOOP;
   CLOSE c_expenses;
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error retrieving expenses: ' || SQLERRM);
      CLOSE c_expenses;
END;
```

TRIGGERS

```
CREATE OR REPLACE TRIGGER Update_Budget_Last_Modified
AFTER INSERT OR UPDATE ON Expenses
FOR EACH ROW
BEGIN
   UPDATE Budgets
   SET last_modified = SYSDATE
   WHERE budget_id = :NEW.budget_id;
END;
```