## **Case Study Documentation**

## **Title**

## **Financial Transaction Management System**

**Group 2- Oracle SQL, PLSQL \_CSD 2024 Selects Batch**

### **Problem Statement:**

### Design and implement a Financial Transaction Management System using Oracle SQL and PL/SQL. The system will be used to manage financial transactions, reconcile accounts, and generate financial reports. Your task is to create the necessary database schema, populate the database with sample data, and develop PL/SQL procedures to handle transaction processing, account reconciliation, and report generation.

**Submitted by : Bheemavarapu Ajay Kumar**

**Superset Id : 5007905**

**Candidate Id: 30743806**

### **1. Introduction**

This document provides a step-by-step guide to setting up and running the Financial Transaction Management System. The system manages financial transactions, reconciles accounts, and generates financial reports using Oracle SQL and PL/SQL.

### **2. Prerequisites**

* Oracle SQL Developer installed on your laptop.
* Oracle Database 21c Express Edition installed and running.
* Access to the Oracle database with the necessary privileges to create tables, insert data, and execute PL/SQL scripts.
* Or you can run this code in Oracle Live Server https://livesql.oracle.com/apex/f?p=590:1000

### **3. Setting Up the Database Schema**

#### **Step 1: Create the Accounts Table**

The Accounts table stores unique account information.

|  |
| --- |
| *-- Create Accounts table to store unique account information*  CREATE TABLE Accounts (      ACCOUNT\_ID NUMBER PRIMARY KEY  *-- We can add more fields related to accounts if needed*  ); |

#### **Step 2: Create the FinancialTransactions Table**

This table manages all financial transactions.

|  |
| --- |
| *-- Create FinancialTransactions table*  CREATE TABLE FinancialTransactions (      TRANSACTION\_ID NUMBER PRIMARY KEY,      ACCOUNT\_ID NUMBER NOT NULL,      TRANSACTION\_DATE DATE NOT NULL,      AMOUNT NUMBER NOT NULL,      DESCRIPTION VARCHAR2(255),      FOREIGN KEY (ACCOUNT\_ID) REFERENCES Accounts(ACCOUNT\_ID)  ); |

#### **Step 3: Create the AccountReconciliations Table**

This table is used to reconcile accounts based on transactions.

|  |
| --- |
| -- Create AccountReconciliations table with foreign key reference to ACCOUNT\_IDCREATE TABLE AccountReconciliations (RECONCILIATION\_ID NUMBER PRIMARY KEY,ACCOUNT\_ID NUMBER NOT NULL,RECONCILIATION\_DATE DATE NOT NULL,BALANCE NUMBER NOT NULL,STATUS VARCHAR2(50),CONSTRAINT fk\_account\_reconciliationFOREIGN KEY (ACCOUNT\_ID) REFERENCES Accounts(ACCOUNT\_ID)); |

#### **Step 4: Create the FinancialReports Table**

This table stores generated financial reports.

|  |
| --- |
| -- Create FinancialReports table  CREATE TABLE FinancialReports (      REPORT\_ID NUMBER PRIMARY KEY,      REPORT\_DATE DATE NOT NULL,      ACCOUNT\_BALANCES CLOB,      TRANSACTION\_SUMMARY CLOB,      RECONCILIATION\_STATUS VARCHAR2(50)  ); |

### **4. Populating the Database with Sample Data**

#### **Step 1: Insert Sample Data into Accounts Table**

|  |
| --- |
| -- Insert sample data into Accounts table  INSERT INTO Accounts (ACCOUNT\_ID) VALUES (101);  INSERT INTO Accounts (ACCOUNT\_ID) VALUES (102);  INSERT INTO Accounts (ACCOUNT\_ID) VALUES (103);  INSERT INTO Accounts (ACCOUNT\_ID) VALUES (104);  INSERT INTO Accounts (ACCOUNT\_ID) VALUES (105);  INSERT INTO Accounts (ACCOUNT\_ID) VALUES (106);  INSERT INTO Accounts (ACCOUNT\_ID) VALUES (107);  INSERT INTO Accounts (ACCOUNT\_ID) VALUES (108); |

**Expected Output:**

|  |
| --- |
|  |

#### **Step 2: Insert Sample Data into FinancialTransactions Table**

Execute the following SQL statements to insert sample data into the FinancialTransactions table:

|  |
| --- |
| -- Insert sample data into FinancialTransactions  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (1, 101, TO\_DATE('2024-08-01', 'YYYY-MM-DD'), 100.00, 'Office supplies purchase');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (2, 102, TO\_DATE('2024-08-02', 'YYYY-MM-DD'), 250.00, 'Client project materials');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (3, 103, TO\_DATE('2024-08-03', 'YYYY-MM-DD'), 150.00, 'Software subscription renewal');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (4, 104, TO\_DATE('2024-08-04', 'YYYY-MM-DD'), 200.00, 'Employee travel reimbursement');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (5, 105, TO\_DATE('2024-08-05', 'YYYY-MM-DD'), 300.00, 'Monthly utilities payment');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (6, 106, TO\_DATE('2024-08-06', 'YYYY-MM-DD'), 120.00, 'Marketing campaign expenses');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (7, 107, TO\_DATE('2024-08-07', 'YYYY-MM-DD'), 180.00, 'Office rent payment');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (8, 108, TO\_DATE('2024-08-08', 'YYYY-MM-DD'), 220.00, 'Hardware repair services');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (9, 101, TO\_DATE('2024-08-09', 'YYYY-MM-DD'), 110.00, 'Client meeting expenses');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (10, 102, TO\_DATE('2024-08-10', 'YYYY-MM-DD'), 260.00, 'IT infrastructure upgrade');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (11, 103, TO\_DATE('2024-08-11', 'YYYY-MM-DD'), 160.00, 'Training program fees');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (12, 104, TO\_DATE('2024-08-12', 'YYYY-MM-DD'), 210.00, 'Conference registration');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (13, 105, TO\_DATE('2024-08-13', 'YYYY-MM-DD'), 310.00, 'New equipment purchase');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (14, 106, TO\_DATE('2024-08-14', 'YYYY-MM-DD'), 130.00, 'Web hosting services');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (15, 107, TO\_DATE('2024-08-15', 'YYYY-MM-DD'), 190.00, 'Annual maintenance contract'); |

**Expected Output:**

|  |
| --- |
|  |

### **5. Developing PL/SQL Procedures**

#### **Procedure 1: Manage Financial Transactions**

This PL/SQL procedure allows you to add, update, or delete transactions.

|  |
| --- |
| ---------------Procedure for Handling Financial Transactions-----------------  CREATE OR REPLACE PROCEDURE ManageTransaction (  p\_transaction\_id IN NUMBER,  p\_account\_id IN NUMBER,  p\_transaction\_date IN DATE,  p\_amount IN NUMBER,  p\_description IN VARCHAR2,  p\_action IN VARCHAR2 -- 'INSERT', 'UPDATE', 'DELETE'  ) AS  BEGIN  IF p\_action = 'INSERT' THEN  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (p\_transaction\_id, p\_account\_id, p\_transaction\_date, p\_amount, p\_description);    ELSIF p\_action = 'UPDATE' THEN  UPDATE FinancialTransactions  SET ACCOUNT\_ID = p\_account\_id,  TRANSACTION\_DATE = p\_transaction\_date,  AMOUNT = p\_amount,  DESCRIPTION = p\_description  WHERE TRANSACTION\_ID = p\_transaction\_id;    ELSIF p\_action = 'DELETE' THEN  DELETE FROM FinancialTransactions  WHERE TRANSACTION\_ID = p\_transaction\_id;    ELSE  RAISE\_APPLICATION\_ERROR(-20001, 'Invalid action specified');  END IF;    COMMIT;  END ManageTransaction;  / |

**Test 1: Add/INSERT a New Transaction**

|  |
| --- |
| --Test 1: Add/INSERT a New Transaction  BEGIN  ManageTransaction(  p\_transaction\_id => 17,  p\_account\_id => 101,  p\_transaction\_date => TO\_DATE('2024-08-16', 'YYYY-MM-DD'),  p\_amount => 300.00,  p\_description => 'Payment for invoice #017',  p\_action => 'INSERT'  );  END;  /  --Check Transaction Where Added--  SELECT \* FROM FinancialTransactions; |

**Expected Output:** New Record where added in the last row of tablesuccessfully.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TRANSACTION\_ID** | **ACCOUNT\_ID** | **TRANSACTION\_DATE** | **AMOUNT** | **DESCRIPTION** |
| 1 | 101 | 01-AUG-24 | 100 | Office supplies purchase |
| 2 | 102 | 02-AUG-24 | 250 | Client project materials |
| 3 | 103 | 03-AUG-24 | 150 | Software subscription renewal |
| 4 | 104 | 04-AUG-24 | 200 | Employee travel reimbursement |
| 5 | 105 | 05-AUG-24 | 300 | Monthly utilities payment |
| 6 | 106 | 06-AUG-24 | 120 | Marketing campaign expenses |
| 7 | 107 | 07-AUG-24 | 180 | Office rent payment |
| 8 | 108 | 08-AUG-24 | 220 | Hardware repair services |
| 9 | 101 | 09-AUG-24 | 110 | Client meeting expenses |
| 10 | 102 | 10-AUG-24 | 260 | IT infrastructure upgrade |
| 11 | 103 | 11-AUG-24 | 160 | Training program fees |
| 12 | 104 | 12-AUG-24 | 210 | Conference registration |
| 13 | 105 | 13-AUG-24 | 310 | New equipment purchase |
| 14 | 106 | 14-AUG-24 | 130 | Web hosting services |
| 15 | 107 | 15-AUG-24 | 190 | Annual maintenance contract |
| 17 | 101 | 16-AUG-24 | 300 | Payment for invoice #017 |

**Test 2: Update an Existing Transaction**

|  |
| --- |
| --Test 2: Update an Existing Transaction  BEGIN  ManageTransaction(  p\_transaction\_id => 1,  p\_account\_id => 101,  p\_transaction\_date => TO\_DATE('2024-08-17', 'YYYY-MM-DD'),  p\_amount => 130.00,  p\_description => 'Updated payment for invoice #001',  p\_action => 'UPDATE'  );  END;  /  --Check Transaction Where Updated--  SELECT \* FROM FinancialTransactions; |

**Expected Output:** Now AMOUNT of ACCOUNT\_ID is UPDATED - 100 to 130 Successfully.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TRANSACTION\_ID** | **ACCOUNT\_ID** | **TRANSACTION\_DATE** | **AMOUNT** | **DESCRIPTION** |
| 1 | 101 | 17-AUG-24 | 130 | Updated payment for invoice #001 |

#### **Test 3: Delete a Transaction**

|  |
| --- |
| --Test 3: Delete a Transaction  BEGIN  ManageTransaction(  p\_transaction\_id => 17,  p\_account\_id => 101,  p\_transaction\_date => NULL,  p\_amount => NULL,  p\_description => NULL,  p\_action => 'DELETE'  );  END;  /  --Check Transaction Where Deleted--  SELECT \* FROM FinancialTransactions; |

**Expected Output:** Transaction id = 17 is successfully deleted.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TRANSACTION\_ID** | **ACCOUNT\_ID** | **TRANSACTION\_DATE** | **AMOUNT** | **DESCRIPTION** |
| 1 | 101 | 17-AUG-24 | 130 | Updated payment for invoice #001 |
| 2 | 102 | 02-AUG-24 | 250 | Client project materials |
| 3 | 103 | 03-AUG-24 | 150 | Software subscription renewal |
| 4 | 104 | 04-AUG-24 | 200 | Employee travel reimbursement |
| 5 | 105 | 05-AUG-24 | 300 | Monthly utilities payment |
| 6 | 106 | 06-AUG-24 | 120 | Marketing campaign expenses |
| 7 | 107 | 07-AUG-24 | 180 | Office rent payment |
| 8 | 108 | 08-AUG-24 | 220 | Hardware repair services |
| 9 | 101 | 09-AUG-24 | 110 | Client meeting expenses |
| 10 | 102 | 10-AUG-24 | 260 | IT infrastructure upgrade |
| 11 | 103 | 11-AUG-24 | 160 | Training program fees |
| 12 | 104 | 12-AUG-24 | 210 | Conference registration |
| 13 | 105 | 13-AUG-24 | 310 | New equipment purchase |
| 14 | 106 | 14-AUG-24 | 130 | Web hosting services |
| 15 | 107 | 15-AUG-24 | 190 | Annual maintenance contract |

#### **Procedure 2: Reconcile Accounts**

This procedure reconciles accounts by calculating balances and updating reconciliation statuses.

|  |
| --- |
| ----------------Procedure for Account Reconciliation-----------------------  --Creating Sequences Account Reconciliation  CREATE SEQUENCE AccountReconciliations\_seq  START WITH 1  INCREMENT BY 1  NOCACHE  NOCYCLE;  /  --Procedure for Account Reconciliation  CREATE OR REPLACE PROCEDURE ReconcileAccount (  p\_account\_id IN NUMBER,  p\_reconciliation\_date IN DATE  ) AS  v\_balance NUMBER;  v\_status VARCHAR2(50);  BEGIN  -- Calculate the balance for the given account  BEGIN  SELECT NVL(SUM(AMOUNT), 0)  INTO v\_balance  FROM FinancialTransactions  WHERE ACCOUNT\_ID = p\_account\_id;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  v\_balance := 0;  WHEN OTHERS THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Error calculating balance: ' || SQLERRM);  END;    -- Determine reconciliation status  v\_status := 'Completed'; -- Example status, adjust as needed    -- Check if a reconciliation record exists for the account  DECLARE  v\_exists NUMBER;  BEGIN  SELECT COUNT(\*)  INTO v\_exists  FROM AccountReconciliations  WHERE ACCOUNT\_ID = p\_account\_id  AND RECONCILIATION\_DATE = p\_reconciliation\_date;    IF v\_exists > 0 THEN  -- Update existing reconciliation record  UPDATE AccountReconciliations  SET BALANCE = v\_balance,  STATUS = v\_status  WHERE ACCOUNT\_ID = p\_account\_id  AND RECONCILIATION\_DATE = p\_reconciliation\_date;  ELSE  -- Insert new reconciliation record  INSERT INTO AccountReconciliations (RECONCILIATION\_ID, ACCOUNT\_ID, RECONCILIATION\_DATE, BALANCE, STATUS)  VALUES (AccountReconciliations\_seq.NEXTVAL, p\_account\_id, p\_reconciliation\_date, v\_balance, v\_status);  END IF;  EXCEPTION  WHEN OTHERS THEN  RAISE\_APPLICATION\_ERROR(-20003, 'Error reconciling account: ' || SQLERRM);  END;    COMMIT;  END ReconcileAccount;  / |

**Testing the Procedure:** To test ReconcileAccount, execute the following:

|  |
| --- |
| --Check Existing Reports Verify if there's already a report with the same REPORT\_ID  SELECT \* FROM AccountReconciliations WHERE ACCOUNT\_ID = 101;  --Test 4: Reconcile an Account  BEGIN  ReconcileAccount(  p\_account\_id => 101,  p\_reconciliation\_date => TO\_DATE('2024-08-31', 'YYYY-MM-DD')  );  END;  /  --Check  SELECT \* FROM AccountReconciliations; |

**Expected Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RECONCILIATION\_ID** | **ACCOUNT\_ID** | **RECONCILIATION\_DATE** | **BALANCE** | **STATUS** |
| 2 | 101 | 31-AUG-24 | 240 | Completed |

#### **Procedure 3: Generate Financial Reports**

This procedure generates financial reports including account balances, transaction summaries, and reconciliation statuses.

|  |
| --- |
| ---------------------------Procedure for Generating Financial Reports---------------------------------  --Sequence for FinancialReports Table  CREATE SEQUENCE FinancialReports\_seq  START WITH 1  INCREMENT BY 1  NOCACHE  NOCYCLE;  /  --Procedure for Generating Financial Reports  CREATE OR REPLACE PROCEDURE GenerateFinancialReport (  p\_report\_date IN DATE  ) AS  v\_account\_balances CLOB;  v\_transaction\_summary CLOB;  v\_reconciliation\_status VARCHAR2(50);  BEGIN  -- Aggregate account balances  SELECT LISTAGG(' Account no ' || ACCOUNT\_ID || ': ' || NVL(SUM(AMOUNT), 0), ' , ') WITHIN GROUP (ORDER BY ACCOUNT\_ID)  INTO v\_account\_balances  FROM FinancialTransactions  GROUP BY ACCOUNT\_ID;  -- Summarize transactions  SELECT 'Total Transactions: ' || COUNT(\*) || ', Total Amount: ' || SUM(AMOUNT)  INTO v\_transaction\_summary  FROM FinancialTransactions;  -- Determine reconciliation status  v\_reconciliation\_status := 'All reconciliations completed'; -- Example status, adjust as needed  -- Insert report into FinancialReports table  INSERT INTO FinancialReports (REPORT\_ID, REPORT\_DATE, ACCOUNT\_BALANCES, TRANSACTION\_SUMMARY, RECONCILIATION\_STATUS)  VALUES (  FinancialReports\_seq.NEXTVAL, -- Ensure FinancialReports\_seq exists  p\_report\_date,  v\_account\_balances,  v\_transaction\_summary,  v\_reconciliation\_status  );  COMMIT;  END GenerateFinancialReport;  / |

**Testing the Procedure:** To test GenerateFinancialReport, execute the following:

|  |
| --- |
| --Test 5: Generate a Financial Report  BEGIN  GenerateFinancialReport(  p\_report\_date => TO\_DATE('2024-08-31', 'YYYY-MM-DD')  );  END;  /  --Verify Results  SELECT \* FROM FinancialReports; |

**Expected Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **REPORT\_ID** | **REPORT\_DATE** | **ACCOUNT\_BALANCES** | **TRANSACTION\_SUMMARY** | **RECONCILIATION\_STATUS** |
| 5 | 31-AUG-24 | Account no 101: 240 ,  Account no 102: 510 ,  Account no 103: 310 ,  Account no 104: 410 ,  Account no 105: 610 ,  Account no 106: 250 ,  Account no 107: 370 ,  Account no 108: 220 | Total Transactions: 15, Total Amount: 2920 | All reconciliations completed |

### **Entire Code**

|  |
| --- |
| -- Create Accounts table to store unique account information  CREATE TABLE Accounts (  ACCOUNT\_ID NUMBER PRIMARY KEY  -- You can add more fields related to accounts if needed  );  -- Create FinancialTransactions table  CREATE TABLE FinancialTransactions (  TRANSACTION\_ID NUMBER PRIMARY KEY,  ACCOUNT\_ID NUMBER NOT NULL,  TRANSACTION\_DATE DATE NOT NULL,  AMOUNT NUMBER NOT NULL,  DESCRIPTION VARCHAR2(255),  FOREIGN KEY (ACCOUNT\_ID) REFERENCES Accounts(ACCOUNT\_ID)  );  -- Create AccountReconciliations table with foreign key reference to ACCOUNT\_ID  CREATE TABLE AccountReconciliations (  RECONCILIATION\_ID NUMBER PRIMARY KEY,  ACCOUNT\_ID NUMBER NOT NULL,  RECONCILIATION\_DATE DATE NOT NULL,  BALANCE NUMBER NOT NULL,  STATUS VARCHAR2(50),  CONSTRAINT fk\_account\_reconciliation  FOREIGN KEY (ACCOUNT\_ID) REFERENCES Accounts(ACCOUNT\_ID)  );  -- Create FinancialReports table  CREATE TABLE FinancialReports (  REPORT\_ID NUMBER PRIMARY KEY,  REPORT\_DATE DATE NOT NULL,  ACCOUNT\_BALANCES CLOB,  TRANSACTION\_SUMMARY CLOB,  RECONCILIATION\_STATUS VARCHAR2(50)  );  -- Insert sample data into Accounts table  INSERT INTO Accounts (ACCOUNT\_ID) VALUES (101);  INSERT INTO Accounts (ACCOUNT\_ID) VALUES (102);  INSERT INTO Accounts (ACCOUNT\_ID) VALUES (103);  INSERT INTO Accounts (ACCOUNT\_ID) VALUES (104);  INSERT INTO Accounts (ACCOUNT\_ID) VALUES (105);  INSERT INTO Accounts (ACCOUNT\_ID) VALUES (106);  INSERT INTO Accounts (ACCOUNT\_ID) VALUES (107);  INSERT INTO Accounts (ACCOUNT\_ID) VALUES (108);  SELECT \* FROM Accounts;  -- Insert sample data into FinancialTransactions  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (1, 101, TO\_DATE('2024-08-01', 'YYYY-MM-DD'), 100.00, 'Office supplies purchase');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (2, 102, TO\_DATE('2024-08-02', 'YYYY-MM-DD'), 250.00, 'Client project materials');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (3, 103, TO\_DATE('2024-08-03', 'YYYY-MM-DD'), 150.00, 'Software subscription renewal');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (4, 104, TO\_DATE('2024-08-04', 'YYYY-MM-DD'), 200.00, 'Employee travel reimbursement');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (5, 105, TO\_DATE('2024-08-05', 'YYYY-MM-DD'), 300.00, 'Monthly utilities payment');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (6, 106, TO\_DATE('2024-08-06', 'YYYY-MM-DD'), 120.00, 'Marketing campaign expenses');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (7, 107, TO\_DATE('2024-08-07', 'YYYY-MM-DD'), 180.00, 'Office rent payment');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (8, 108, TO\_DATE('2024-08-08', 'YYYY-MM-DD'), 220.00, 'Hardware repair services');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (9, 101, TO\_DATE('2024-08-09', 'YYYY-MM-DD'), 110.00, 'Client meeting expenses');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (10, 102, TO\_DATE('2024-08-10', 'YYYY-MM-DD'), 260.00, 'IT infrastructure upgrade');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (11, 103, TO\_DATE('2024-08-11', 'YYYY-MM-DD'), 160.00, 'Training program fees');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (12, 104, TO\_DATE('2024-08-12', 'YYYY-MM-DD'), 210.00, 'Conference registration');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (13, 105, TO\_DATE('2024-08-13', 'YYYY-MM-DD'), 310.00, 'New equipment purchase');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (14, 106, TO\_DATE('2024-08-14', 'YYYY-MM-DD'), 130.00, 'Web hosting services');  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (15, 107, TO\_DATE('2024-08-15', 'YYYY-MM-DD'), 190.00, 'Annual maintenance contract');  SELECT \* FROM FinancialTransactions;  --\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-----  ----------------------Procedure for Handling Financial Transactions---------------------  CREATE OR REPLACE PROCEDURE ManageTransaction (  p\_transaction\_id IN NUMBER,  p\_account\_id IN NUMBER,  p\_transaction\_date IN DATE,  p\_amount IN NUMBER,  p\_description IN VARCHAR2,  p\_action IN VARCHAR2 -- 'INSERT', 'UPDATE', 'DELETE'  ) AS  BEGIN  IF p\_action = 'INSERT' THEN  INSERT INTO FinancialTransactions (TRANSACTION\_ID, ACCOUNT\_ID, TRANSACTION\_DATE, AMOUNT, DESCRIPTION)  VALUES (p\_transaction\_id, p\_account\_id, p\_transaction\_date, p\_amount, p\_description);    ELSIF p\_action = 'UPDATE' THEN  UPDATE FinancialTransactions  SET ACCOUNT\_ID = p\_account\_id,  TRANSACTION\_DATE = p\_transaction\_date,  AMOUNT = p\_amount,  DESCRIPTION = p\_description  WHERE TRANSACTION\_ID = p\_transaction\_id;    ELSIF p\_action = 'DELETE' THEN  DELETE FROM FinancialTransactions  WHERE TRANSACTION\_ID = p\_transaction\_id;    ELSE  RAISE\_APPLICATION\_ERROR(-20001, 'Invalid action specified');  END IF;    COMMIT;  END ManageTransaction;  /  --Testing the Procedures  --Test 1: Add/INSERT a New Transaction  BEGIN  ManageTransaction(  p\_transaction\_id => 17,  p\_account\_id => 101,  p\_transaction\_date => TO\_DATE('2024-08-16', 'YYYY-MM-DD'),  p\_amount => 300.00,  p\_description => 'Payment for invoice #017',  p\_action => 'INSERT'  );  END;  /  --Check Transaction Where Added--  SELECT \* FROM FinancialTransactions;  --Test 2: Update an Existing Transaction  BEGIN  ManageTransaction(  p\_transaction\_id => 1,  p\_account\_id => 101,  p\_transaction\_date => TO\_DATE('2024-08-17', 'YYYY-MM-DD'),  p\_amount => 130.00,  p\_description => 'Updated payment for invoice #001',  p\_action => 'UPDATE'  );  END;  /  --Check Transaction Where Updated--  SELECT \* FROM FinancialTransactions;  --Test 3: Delete a Transaction  BEGIN  ManageTransaction(  p\_transaction\_id => 17,  p\_account\_id => 101,  p\_transaction\_date => NULL,  p\_amount => NULL,  p\_description => NULL,  p\_action => 'DELETE'  );  END;  /  --Check Transaction Where Deleted--  SELECT \* FROM FinancialTransactions;  ----------------------------Procedure for Account Reconciliation-----------------------------  --Creating Sequences Account Reconciliation  CREATE SEQUENCE AccountReconciliations\_seq  START WITH 1  INCREMENT BY 1  NOCACHE  NOCYCLE;  /  --Procedure for Account Reconciliation  CREATE OR REPLACE PROCEDURE ReconcileAccount (  p\_account\_id IN NUMBER,  p\_reconciliation\_date IN DATE  ) AS  v\_balance NUMBER;  v\_status VARCHAR2(50);  BEGIN  -- Calculate the balance for the given account  BEGIN  SELECT NVL(SUM(AMOUNT), 0)  INTO v\_balance  FROM FinancialTransactions  WHERE ACCOUNT\_ID = p\_account\_id;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  v\_balance := 0;  WHEN OTHERS THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Error calculating balance: ' || SQLERRM);  END;    -- Determine reconciliation status  v\_status := 'Completed'; -- Example status, adjust as needed    -- Check if a reconciliation record exists for the account  DECLARE  v\_exists NUMBER;  BEGIN  SELECT COUNT(\*)  INTO v\_exists  FROM AccountReconciliations  WHERE ACCOUNT\_ID = p\_account\_id  AND RECONCILIATION\_DATE = p\_reconciliation\_date;    IF v\_exists > 0 THEN  -- Update existing reconciliation record  UPDATE AccountReconciliations  SET BALANCE = v\_balance,  STATUS = v\_status  WHERE ACCOUNT\_ID = p\_account\_id  AND RECONCILIATION\_DATE = p\_reconciliation\_date;  ELSE  -- Insert new reconciliation record  INSERT INTO AccountReconciliations (RECONCILIATION\_ID, ACCOUNT\_ID, RECONCILIATION\_DATE, BALANCE, STATUS)  VALUES (AccountReconciliations\_seq.NEXTVAL, p\_account\_id, p\_reconciliation\_date, v\_balance, v\_status);  END IF;  EXCEPTION  WHEN OTHERS THEN  RAISE\_APPLICATION\_ERROR(-20003, 'Error reconciling account: ' || SQLERRM);  END;    COMMIT;  END ReconcileAccount;  /  --Check Existing Reports Verify if there's already a report with the same REPORT\_ID  SELECT \* FROM AccountReconciliations WHERE ACCOUNT\_ID = 101;  --Test 4: Reconcile an Account  BEGIN  ReconcileAccount(  p\_account\_id => 101,  p\_reconciliation\_date => TO\_DATE('2024-08-31', 'YYYY-MM-DD')  );  END;  /  --Check  SELECT \* FROM AccountReconciliations;  ---------------------------Procedure for Generating Financial Reports---------------------------------  --Sequence for FinancialReports Table  CREATE SEQUENCE FinancialReports\_seq  START WITH 1  INCREMENT BY 1  NOCACHE  NOCYCLE;  /  --Procedure for Generating Financial Reports  CREATE OR REPLACE PROCEDURE GenerateFinancialReport (  p\_report\_date IN DATE  ) AS  v\_account\_balances CLOB;  v\_transaction\_summary CLOB;  v\_reconciliation\_status VARCHAR2(50);  BEGIN  -- Aggregate account balances  SELECT LISTAGG(' Account no ' || ACCOUNT\_ID || ': ' || NVL(SUM(AMOUNT), 0), ' , ') WITHIN GROUP (ORDER BY ACCOUNT\_ID)  INTO v\_account\_balances  FROM FinancialTransactions  GROUP BY ACCOUNT\_ID;  -- Summarize transactions  SELECT 'Total Transactions: ' || COUNT(\*) || ', Total Amount: ' || SUM(AMOUNT)  INTO v\_transaction\_summary  FROM FinancialTransactions;  -- Determine reconciliation status  v\_reconciliation\_status := 'All reconciliations completed'; -- Example status, adjust as needed  -- Insert report into FinancialReports table  INSERT INTO FinancialReports (REPORT\_ID, REPORT\_DATE, ACCOUNT\_BALANCES, TRANSACTION\_SUMMARY, RECONCILIATION\_STATUS)  VALUES (  FinancialReports\_seq.NEXTVAL, -- Ensure FinancialReports\_seq exists  p\_report\_date,  v\_account\_balances,  v\_transaction\_summary,  v\_reconciliation\_status  );  COMMIT;  END GenerateFinancialReport;  /  --Test 5: Generate a Financial Report  BEGIN  GenerateFinancialReport(  p\_report\_date => TO\_DATE('2024-08-31', 'YYYY-MM-DD')  );  END;  /  --Verify Results  SELECT \* FROM FinancialReports; |