**WEEK-02 HANDS ON SOLUTIONS**

**PL/SQL EXERCISE SOLUTIONS**

**Exercise 3: Stored Procedures**

Scenario 1: The bank needs to process monthly interest for all savings accounts.

o Question: Write a stored procedure ProcessMonthlyInterest that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

Scenario 2: The bank wants to implement a bonus scheme for employees based on their performance.

o Question: Write a stored procedure UpdateEmployeeBonus that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

Scenario 3: Customers should be able to transfer funds between their accounts.

o Question: Write a stored procedure TransferFunds that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

**EXPLANATION:**

In this exercise, PL/SQL stored procedures are used to simulate core banking operations such as interest processing, bonus calculation, and fund transfers.

To support these operations, I created three tables: savings\_accounts, employees, and accounts.

Each table is designed with attributes that reflect real-world data structures used in banking systems.

The savings\_accounts table contains three fields: account\_id (primary key), customer\_id, and balance.

This table stores information about individual customer savings accounts. The first stored procedure, ProcessMonthlyInterest, operates on this table by applying a fixed 1% interest to each balance using a simple UPDATE query.

This simulates monthly interest being credited to every customer's savings account.

The second table, employees, includes attributes such as emp\_id (primary key), name, department\_id, and salary.

It represents the bank's employee database. The procedure UpdateEmployeeBonus uses this table to provide performance-based salary increments.

It takes a department ID and bonus percentage as parameters and increases the salary of every employee in that department accordingly.

Lastly, the accounts table is used to simulate a general-purpose customer account system, which is useful for fund transfers.

It includes account\_id (primary key), customer\_id, and balance.

The TransferFunds procedure operates on this table to move money from one account to another.

Before performing the transfer, it checks whether the source account has sufficient funds.

If the balance is insufficient, the procedure raises an application error; otherwise, it updates both accounts accordingly.

Each stored procedure performs database updates without requiring user input/output during execution.

The correctness of the procedure is verified by executing SELECT queries after running the procedures, which confirm that interest is applied, bonuses are added, and funds are successfully transferred between accounts.

This exercise effectively demonstrates how PL/SQL procedures can be used to implement modular, secure, and reusable components in a banking database application.

**OUTPUT**











