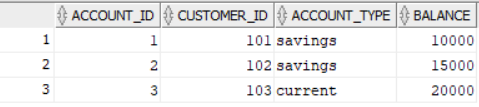
**Exercise 3: Stored Procedures**

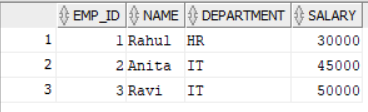
For this task we need a pre-defined data for which we use file named as ‘Data\_Insertion.sql’.

Data Output –

SELECT \* FROM accounts;



SELECT \* FROM employees;



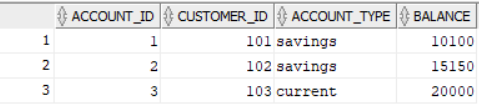
**Scenario 1: Process Monthly Interest**

* Objective: Apply a 1% monthly interest to the balance of all savings accounts.
* Logic: Use a stored procedure that updates all rows in the accounts table where account\_type = 'savings' by adding 1% of the current balance.
* The Respective Code is named as with file ‘Monthly\_Interest.sql’.

Output –

EXEC ProcessMonthlyInterest;

SELECT \* FROM accounts;



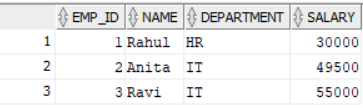
**Scenario 2: Update Employee Bonus**

* Objective: Increase the salary of employees in a specific department by a bonus percentage passed as a parameter.
* Logic: Use a stored procedure with two parameters (department name and bonus %) and apply bonus using a simple UPDATE based on department match.
* The Respective Code is named as with file ‘Add\_Bonus.sql’.

Output –

EXEC UpdateEmployeeBonus;

SELECT \* FROM employees;



**Scenario 3: Transfer Funds Between Accounts**

* Objective: Transfer a specific amount from one account to another, only if the sender has sufficient balance.
* Logic: Use a stored procedure that:
  1. Retrieves the current balance of the sender.
  2. Checks if it's sufficient.
  3. Deducts from sender and adds to receiver if valid.
  4. Otherwise, raises an exception and prints a message.
* The Respective Code is named as with file ‘Transfer\_Funds.sql’.

Output –

EXEC TransferFunds;

SELECT \* FROM accounts;

