Stored Procedures:

Database Initialization:

```
CREATE TABLE SavingsAccounts (
   account_id INT PRIMARY KEY,
   customer_id INT,
   balance DECIMAL(15, 2),
   account_type VARCHAR(20) CHECK (account_type = 'SAVINGS')
);

CREATE TABLE Employees (
   employee_id INT PRIMARY KEY,
   name VARCHAR(100),
   department VARCHAR(50),
   salary DECIMAL(15, 2)
);

CREATE TABLE Accounts (
   account_id INT PRIMARY KEY,
   customer_id INT,
   balance DECIMAL(15, 2),
   account_type VARCHAR(20)
);

INSERT ALL
  INTO SavingsAccounts (account_id, customer_id, balance, account_type) VALUES
(101, 1, 5000.00, 'SAVINGS')
  INTO SavingsAccounts (account_id, customer_id, balance, account_type) VALUES
(102, 2, 12000.00, 'SAVINGS')
  INTO SavingsAccounts (account_id, customer_id, balance, account_type) VALUES
(103, 3, 7500.00, 'SAVINGS')
  INTO SavingsAccounts (account_id, customer_id, balance, account_type) VALUES
(104, 4, 3000.00, 'SAVINGS')
  INTO SavingsAccounts (account_id, customer_id, balance, account_type) VALUES
(105, 5, 15000.00, 'SAVINGS')
SELECT * FROM dual;
```

```
INSERT ALL
  INTO Employees (employee_id, name, department, salary) VALUES (1, 'Manan', 'IT',
75000.00)
  INTO Employees (employee_id, name, department, salary) VALUES (2, 'Saswat', 'HR',
65000.00)
  INTO Employees (employee_id, name, department, salary) VALUES (3, 'Kinshuk',
'Finance', 82000.00)
  INTO Employees (employee_id, name, department, salary) VALUES (4, 'Ayush', 'IT',
78000.00)
  INTO Employees (employee_id, name, department, salary) VALUES (5, 'Hrishi',
'Marketing', 70000.00)
  INTO Employees (employee_id, name, department, salary) VALUES (6, 'Arvin',
'Finance', 85000.00)
  INTO Employees (employee_id, name, department, salary) VALUES (7, 'Saras', 'HR',
60000.00)
SELECT * FROM dual;


INSERT ALL
  INTO Accounts (account_id, customer_id, balance, account_type) VALUES (1001, 1,
5000.00, 'CHECKING')
  INTO Accounts (account_id, customer_id, balance, account_type) VALUES (1002, 1,
3000.00, 'SAVINGS')
  INTO Accounts (account_id, customer_id, balance, account_type) VALUES (1003, 2,
12000.00, 'CHECKING')
  INTO Accounts (account_id, customer_id, balance, account_type) VALUES (1004, 2,
5000.00, 'SAVINGS')
  INTO Accounts (account_id, customer_id, balance, account_type) VALUES (1005, 3,
7500.00, 'CHECKING')
  INTO Accounts (account_id, customer_id, balance, account_type) VALUES (1006, 3,
2000.00, 'SAVINGS')
  INTO Accounts (account_id, customer_id, balance, account_type) VALUES (1007, 4,
3000.00, 'CHECKING')
  INTO Accounts (account_id, customer_id, balance, account_type) VALUES (1008, 4,
10000.00, 'SAVINGS')
```

```
  INTO Accounts (account_id, customer_id, balance, account_type) VALUES (1009, 5, 
15000.00, 'CHECKING')
  INTO Accounts (account_id, customer_id, balance, account_type) VALUES (1010, 5, 
50000.00, 'SAVINGS')
SELECT * FROM dual;
```

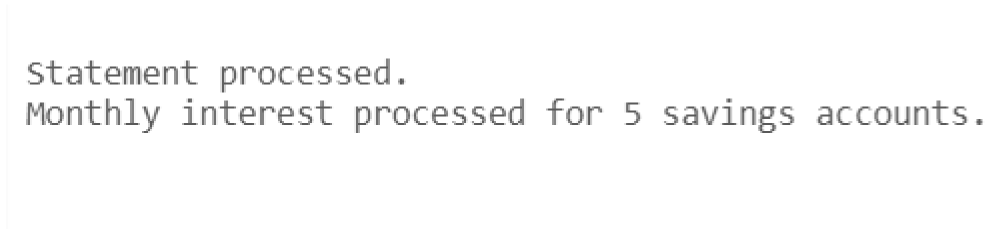
Scenario 1:

Code:

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS
  v_count NUMBER;
BEGIN
  -- Update balance for all savings accounts with 1% interest
  UPDATE SavingsAccounts
  SET balance = balance * 1.01
  WHERE account_type = 'SAVINGS';

  v_count := SQL%ROWCOUNT;
  DBMS_OUTPUT.PUT_LINE('Monthly interest processed for ' || v_count || '
savings accounts.');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error processing monthly interest: ' ||
SQLERRM);
END;
/
```

Output:



```
Statement processed.
Monthly interest processed for 5 savings accounts.
```

Scenario 2:

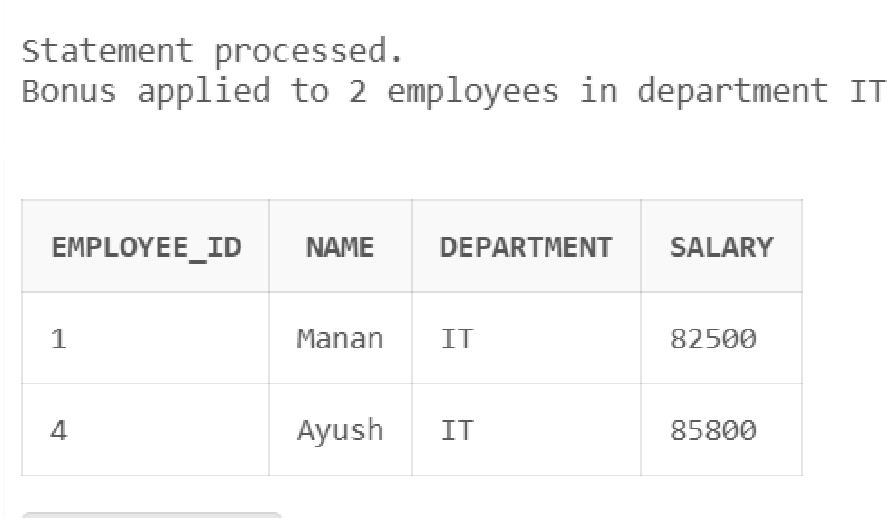
Code:

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(
  dept_name IN VARCHAR2,
  bonus_percentage IN NUMBER
) AS
  v_count NUMBER;
BEGIN

  IF bonus_percentage < 0 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Bonus percentage cannot be
negative');
  ELSE
    UPDATE Employees
    SET salary = salary * (1 + bonus_percentage / 100)
    WHERE department = dept_name;

    v_count := SQL%ROWCOUNT;
    DBMS_OUTPUT.PUT_LINE('Bonus applied to ' || v_count || ' employees in
department ' || dept_name);
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error updating employee bonuses: ' ||
SQLERRM);
END;
/
BEGIN
  UpdateEmployeeBonus('IT', 10);
END;
SELECT * FROM Employees WHERE department = 'IT';
```

Output:



```
Statement processed.
Bonus applied to 2 employees in department IT
```

EMPLOYEE_ID	NAME	DEPARTMENT	SALARY
1	Manan	IT	82500
4	Ayush	IT	85800

Scenario 3:

Code:

```
CREATE OR REPLACE PROCEDURE TransferFunds(
  source_account_id IN NUMBER,
  target_account_id IN NUMBER,
  amount IN NUMBER,
  status_message OUT VARCHAR2
) AS
  source_balance NUMBER;
  target_exists NUMBER;
BEGIN

  BEGIN
    SELECT balance INTO source_balance
    FROM Accounts
    WHERE account_id = source_account_id
    FOR UPDATE;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      status_message := 'Source account not found';
      RETURN;
  END;

  IF source_balance < amount THEN
    status_message := 'Insufficient funds in source account';
    RETURN;
  END IF;

  SELECT COUNT(*) INTO target_exists
  FROM Accounts
  WHERE account_id = target_account_id;

  IF target_exists = 0 THEN
```

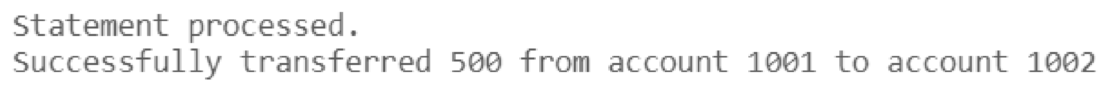
```
    status_message := 'Target account not found';
    RETURN;
  END IF;

  UPDATE Accounts SET balance = balance - amount
  WHERE account_id = source_account_id;

  UPDATE Accounts SET balance = balance + amount
  WHERE account_id = target_account_id;

  COMMIT;
  status_message := 'Successfully transferred ' || amount ||
                 ' from account ' || source_account_id ||
                 ' to account ' || target_account_id;
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
    status_message := 'Error occurred during transfer: ' || SQLERRM;
END;
/
```

Output:



```
Statement processed.
Successfully transferred 500 from account 1001 to account 1002
```