**Exercise 1: Control Structures**

**PLSQL code:**

|  |
| --- |
| use cognizant\_dn;  CREATE TABLE customers (  customer\_id INT PRIMARY KEY,  name VARCHAR(100),  age INT,  balance DECIMAL(12,2),  isVIP BOOLEAN DEFAULT FALSE  );  CREATE TABLE loans (  loan\_id INT PRIMARY KEY,  customer\_id INT,  interest\_rate DECIMAL(5,2), -- Example: 9.25%  due\_date DATE,  FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)  );  INSERT INTO customers VALUES (1, 'Alice', 65, 12000, FALSE);  INSERT INTO customers VALUES (2, 'Bob', 45, 8000, FALSE);  INSERT INTO customers VALUES (3, 'Carol', 70, 5000, FALSE);  INSERT INTO loans VALUES (101, 1, 9.50, CURDATE() + INTERVAL 10 DAY);  INSERT INTO loans VALUES (102, 2, 10.00, CURDATE() + INTERVAL 35 DAY);  INSERT INTO loans VALUES (103, 3, 11.00, CURDATE() + INTERVAL 5 DAY);  UPDATE loans  JOIN customers ON loans.customer\_id = customers.customer\_id  SET loans.interest\_rate = loans.interest\_rate - 1.00  WHERE customers.age > 60;  UPDATE customers  SET isVIP = TRUE  WHERE balance > 10000;  SELECT  c.name,  l.loan\_id,  l.due\_date,  CONCAT('Reminder: Loan ', l.loan\_id, ' for customer ', c.name, ' is due on ', l.due\_date) AS reminder\_message  FROM loans l  JOIN customers c ON c.customer\_id = l.customer\_id  WHERE l.due\_date BETWEEN CURDATE() AND CURDATE() + INTERVAL 30 DAY; |

**Output Screenshots:**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

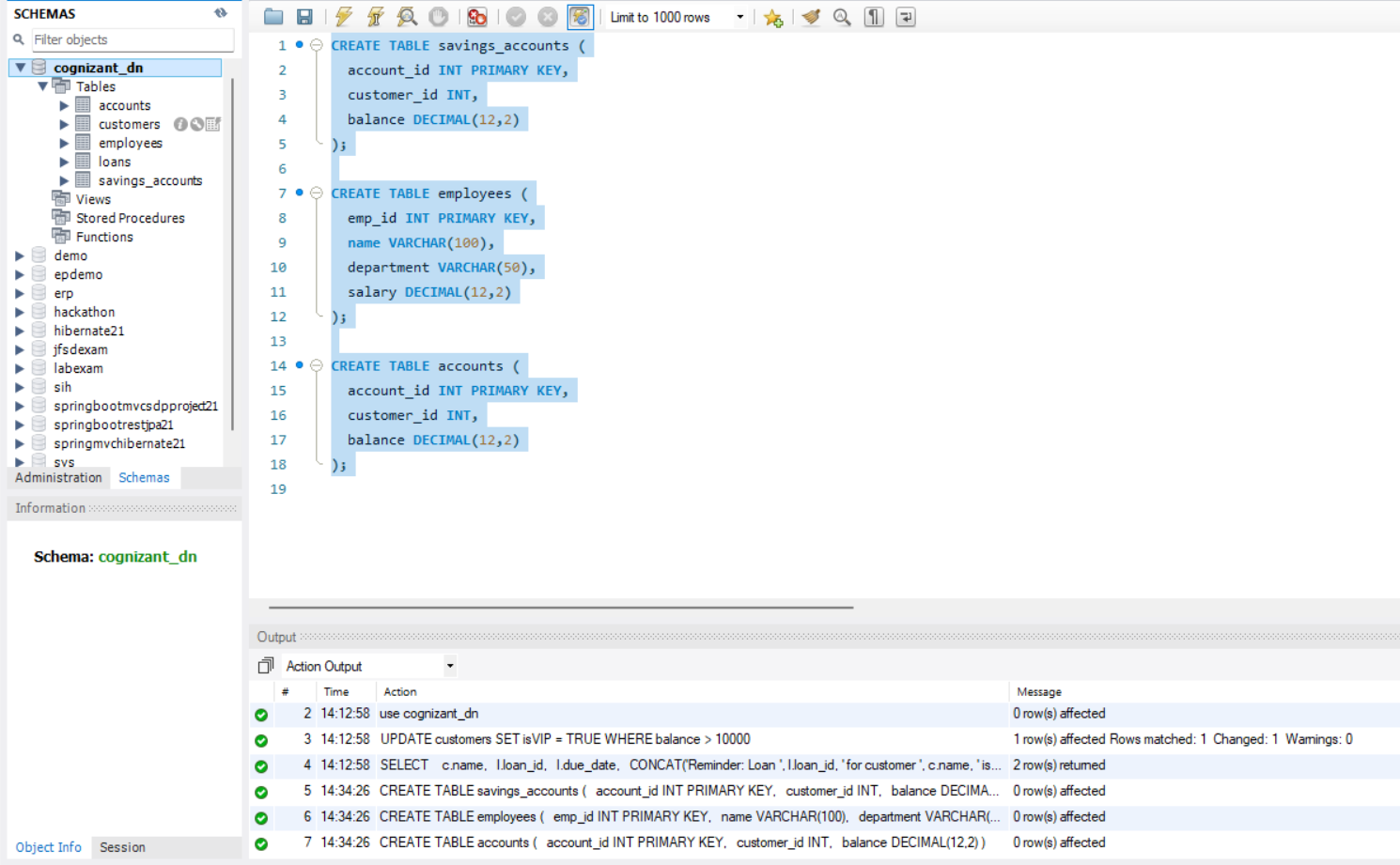
AI-generated content may be incorrect.

**Exercise 3: Stored Procedures**

**PLSQL:**

|  |
| --- |
| CREATE TABLE savings\_accounts (  account\_id INT PRIMARY KEY,  customer\_id INT,  balance DECIMAL(12,2)  );  CREATE TABLE employees (  emp\_id INT PRIMARY KEY,  name VARCHAR(100),  department VARCHAR(50),  salary DECIMAL(12,2)  );  CREATE TABLE accounts (  account\_id INT PRIMARY KEY,  customer\_id INT,  balance DECIMAL(12,2)  );  DELIMITER $$  CREATE PROCEDURE ProcessMonthlyInterest()  BEGIN  UPDATE savings\_accounts  SET balance = balance + (balance \* 0.01);  END $$  DELIMITER ;  DELIMITER $$  CREATE PROCEDURE UpdateEmployeeBonus(  IN dept\_name VARCHAR(50),  IN bonus\_percent DECIMAL(5,2)  )  BEGIN  UPDATE employees  SET salary = salary + (salary \* bonus\_percent / 100)  WHERE department = dept\_name;  END $$  DELIMITER ;  DELIMITER $$  CREATE PROCEDURE TransferFunds(  IN from\_account INT,  IN to\_account INT,  IN transfer\_amount DECIMAL(12,2)  )  BEGIN  DECLARE from\_balance DECIMAL(12,2);  SELECT balance INTO from\_balance  FROM accounts  WHERE account\_id = from\_account;  IF from\_balance >= transfer\_amount THEN    UPDATE accounts  SET balance = balance - transfer\_amount  WHERE account\_id = from\_account;  UPDATE accounts  SET balance = balance + transfer\_amount  WHERE account\_id = to\_account;  ELSE  SIGNAL SQLSTATE '45000'  SET MESSAGE\_TEXT = 'Insufficient balance in source account';  END IF;  END $$  DELIMITER ;  CALL UpdateEmployeeBonus('IT', 10);  CALL TransferFunds(101, 102, 500); |

Output:



A screenshot of a computer

AI-generated content may be incorrect.