```
SQL> CREATE OR REPLACE FUNCTION CalculateAge(
  2      p_dob DATE
  3  ) RETURN NUMBER IS
  4      v_age NUMBER;
  5  BEGIN
  6      -- Calculate age as the difference in years between the current date and DOB
  7      v_age := EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM p_dob);
  8
  9      -- Adjust if the current date is before the birthday this year
 10      IF TO_CHAR(SYSDATE, 'MMDD') < TO_CHAR(p_dob, 'MMDD') THEN
 11          v_age := v_age - 1;
 12      END IF;
 13
 14      RETURN v_age;
 15  END CalculateAge;
 16  /

Function created.

SQL> CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment(
  2      p_loan_amount NUMBER,
  3      p_interest_rate NUMBER,
  4      p_duration_years NUMBER
  5  ) RETURN NUMBER IS
  6      v_monthly_rate NUMBER;
  7      v_months NUMBER;
  8      v_monthly_installment NUMBER;
  9  BEGIN
 10      -- Convert annual interest rate to a monthly rate
 11      v_monthly_rate := p_interest_rate / 100 / 12;
 12
 13      -- Calculate total number of months
 14      v_months := p_duration_years * 12;
 15
 16      -- Calculate monthly installment using the formula for annuity
 17      v_monthly_installment := p_loan_amount * (v_monthly_rate / (1 - POWER(1 + v_monthly_rate, -
v_months)));
 18
 19      RETURN v_monthly_installment;
 20  END CalculateMonthlyInstallment;
 21  /

Function created.
```

```
SQL> CREATE OR REPLACE FUNCTION HasSufficientBalance(
  2      p_account_id NUMBER,
  3      p_amount NUMBER
  4  ) RETURN BOOLEAN IS
  5      v_balance NUMBER;
  6  BEGIN
  7      -- Retrieve the account balance
  8      SELECT Balance INTO v_balance FROM Accounts
  9      WHERE AccountID = p_account_id;
 10
 11      -- Check if the balance is sufficient
 12      RETURN v_balance >= p_amount;
 13  EXCEPTION
 14      WHEN NO_DATA_FOUND THEN
 15          RETURN FALSE;
 16  END HasSufficientBalance;
 17  /

Function created.
```