Exercise 01

```
DECLARE
  v_company_name VARCHAR2(10) := 'IBM';
  v_stock_price NUMBER(10,2);
BEGIN
  SELECT stock_price
  INTO v_stock_price
  FROM stocks
  WHERE company_name = v_company_name
    AND stock_date = TRUNC(SYSDATE);

  DBMS_OUTPUT.PUT_LINE('The current stock price for ' || v_company_name || ' is ' || v_stock_price);
END;
```

Exercise 02

```
DECLARE
  v_company_name VARCHAR2(10) := 'IBM';
  v_stock_price NUMBER(10,2);
BEGIN
  SELECT stock_price
  INTO v_stock_price
  FROM stocks
  WHERE company_name = v_company_name
    AND stock_date = TRUNC(SYSDATE);

  IF v_stock_price < 45 THEN
    DBMS_OUTPUT.PUT_LINE('Current price is very low!');
  ELSIF v_stock_price < 55 THEN
    DBMS_OUTPUT.PUT_LINE('Current price is low!');
  ELSIF v_stock_price < 65 THEN
    DBMS_OUTPUT.PUT_LINE('Current price is medium!');
  ELSIF v_stock_price < 75 THEN
    DBMS_OUTPUT.PUT_LINE('Current price is medium high!');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Current price is high!');
  END IF;
END;
```

Exercise 03

```
DECLARE
 i INTEGER := 9;
BEGIN
 -- loop using FOR loop
 FOR j IN REVERSE 1..9 LOOP
  FOR k IN 1..i LOOP
   DBMS_OUTPUT.PUT(i || ' ');
  END LOOP;
  i := i - 1;
  DBMS_OUTPUT.NEW_LINE;
 END LOOP;

 i := 9;

 -- loop using WHILE loop
 WHILE i >= 1 LOOP
  j := 1;
  WHILE j <= i LOOP
   DBMS_OUTPUT.PUT(i || ' ');
   j := j + 1;
  END LOOP;
  i := i - 1;
  DBMS_OUTPUT.NEW_LINE;
 END LOOP;

 i := 9;

 -- loop using LOOP-EXIT WHEN loop
 LOOP
  FOR j IN 1..i LOOP
   DBMS_OUTPUT.PUT(i || ' ');
  END LOOP;
  i := i - 1;
  DBMS_OUTPUT.NEW_LINE;
  EXIT WHEN i < 1;
 END LOOP;
END;
```

Exercise 04

```
DECLARE
  CURSOR c1 IS
    SELECT purchase_id, client_id, company_id, purchase_date, quantity
    FROM purchase
    WHERE purchase_date < TO_DATE('2000-01-01', 'YYYY-MM-DD')
    FOR UPDATE;

  CURSOR c2 IS
    SELECT purchase_id, client_id, company_id, purchase_date, quantity
    FROM purchase
    WHERE purchase_date >= TO_DATE('2000-01-01', 'YYYY-MM-DD')
    AND purchase_date < TO_DATE('2001-01-01', 'YYYY-MM-DD')
    FOR UPDATE;

  CURSOR c3 IS
    SELECT purchase_id, client_id, company_id, purchase_date, quantity
    FROM purchase
    WHERE purchase_date >= TO_DATE('2001-01-01', 'YYYY-MM-DD')
    AND purchase_date < TO_DATE('2002-01-01', 'YYYY-MM-DD')
    FOR UPDATE;

  bonus_qty INTEGER;
BEGIN
  bonus_qty := 150; -- bonus for purchases made before 1st January 2000

  FOR rec IN c1 LOOP
    UPDATE purchase
    SET quantity = quantity + bonus_qty
    WHERE CURRENT OF c1;
  END LOOP;

  bonus_qty := 100; -- bonus for purchases made before 1st January 2001

  FOR rec IN c2 LOOP
    UPDATE purchase
    SET quantity = quantity + bonus_qty
    WHERE CURRENT OF c2;
  END LOOP;

  bonus_qty := 50; -- bonus for purchases made before 1st January 2002

  FOR rec IN c3 LOOP
    UPDATE purchase
    SET quantity = quantity + bonus_qty
    WHERE CURRENT OF c3;
  END LOOP;

  COMMIT;
END;
```

Exercise 05

```
DECLARE
  CURSOR c_purchase IS
    SELECT purchase_id, client_id, company_id, purchase_date, quantity
    FROM purchase
    FOR UPDATE;

  bonus_qty INTEGER;
  rec_purchase c_purchase%ROWTYPE;
BEGIN
  bonus_qty := 150; -- bonus for purchases made before 1st January 2000

  OPEN c_purchase;
  LOOP
    FETCH c_purchase INTO rec_purchase;
    EXIT WHEN c_purchase%NOTFOUND;

    IF rec_purchase.purchase_date < TO_DATE('2000-01-01', 'YYYY-MM-DD') THEN
      UPDATE purchase
      SET quantity = quantity + bonus_qty
      WHERE CURRENT OF c_purchase;
    END IF;
  END LOOP;
  CLOSE c_purchase;

  bonus_qty := 100; -- bonus for purchases made before 1st January 2001

  OPEN c_purchase;
  LOOP
    FETCH c_purchase INTO rec_purchase;
    EXIT WHEN c_purchase%NOTFOUND;

    IF rec_purchase.purchase_date >= TO_DATE('2000-01-01', 'YYYY-MM-DD')
    AND rec_purchase.purchase_date < TO_DATE('2001-01-01', 'YYYY-MM-DD') THEN
      UPDATE purchase
      SET quantity = quantity + bonus_qty
      WHERE CURRENT OF c_purchase;
    END IF;
  END LOOP;
  CLOSE c_purchase;

  bonus_qty := 50; -- bonus for purchases made before 1st January 2002

  OPEN c_purchase;
  LOOP
    FETCH c_purchase INTO rec_purchase;
    EXIT WHEN c_purchase%NOTFOUND;

    IF rec_purchase.purchase_date >= TO_DATE('2001-01-01', 'YYYY-MM-DD')
    AND rec_purchase.purchase_date < TO_DATE('2002-01-01', 'YYYY-MM-DD') THEN
      UPDATE purchase
      SET quantity = quantity + bonus_qty
      WHERE CURRENT OF c_purchase;
    END IF;
  END LOOP;
  CLOSE c_purchase;

  COMMIT;
END;
```

Anith labsheet eke

Exercise 01

```
CREATE OR REPLACE PROCEDURE UpdateCourse(name_in IN VARCHAR2) IS
  course_number NUMBER;
BEGIN
  -- Look up the course number based on the course name
  SELECT course_number INTO course_number
  FROM courses
  WHERE course_name = name_in;

  -- If no match is found, default the course number to 10000
  IF course_number IS NULL THEN
    course_number := 10000;
  END IF;

  -- Insert a new record into the student_courses table
  INSERT INTO student_courses (course_number)
  VALUES (course_number);

  COMMIT;

  DBMS_OUTPUT.PUT_LINE('New course added: ' || name_in || ' (' || course_number || ')');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLCODE || ' - ' || SQLERRM);
END;
/


  -- To drop the procedure
DROP PROCEDURE UpdateCourse;
```

Exercise 02

```
CREATE OR REPLACE TRIGGER customers_salary_diff
AFTER INSERT OR UPDATE OR DELETE ON customers
FOR EACH ROW
DECLARE
  old_salary customers.salary%TYPE;
  new_salary customers.salary%TYPE;
BEGIN
  IF INSERTING THEN
    old_salary := NULL;
    new_salary := :new.salary;
  ELSIF UPDATING THEN
    old_salary := :old.salary;
    new_salary := :new.salary;
  ELSE -- DELETING
    old_salary := :old.salary;
    new_salary := NULL;
  END IF;

  IF old_salary IS NOT NULL AND new_salary IS NOT NULL THEN
    DBMS_OUTPUT.PUT_LINE('Salary difference for customer ' || :old.id || ': ' || (new_salary - old_salary));
  ELSIF old_salary IS NULL THEN
    DBMS_OUTPUT.PUT_LINE('New customer added: ' || :new.name);
  ELSE -- new_salary IS NULL
    DBMS_OUTPUT.PUT_LINE('Customer ' || :old.id || ' deleted');
  END IF;
END;
/


DROP TRIGGER customers_salary_diff;
```

Exercise 03

a)
```
DECLARE
 v_years_of_service NUMBER;
 v_increment_pct NUMBER;
BEGIN
 FOR emp IN (SELECT emp_id, name, hire_date, salary FROM employees) LOOP
  v_years_of_service := MONTHS_BETWEEN(SYSDATE, emp.hire_date) / 12;
  IF v_years_of_service < 5 THEN
   v_increment_pct := 5;
  ELSIF v_years_of_service < 10 THEN
   v_increment_pct := 10;
  ELSE
   v_increment_pct := 15;
  END IF;
  DBMS_OUTPUT.PUT_LINE(emp.name || ': ' || ROUND(emp.salary * (1 + v_increment_pct/100)));
 END LOOP;
END;
```

b)
```
DECLARE
 CURSOR c_emp IS
  SELECT emp_id, name, job_id, hire_date
  FROM employees
  WHERE hire_date = (SELECT MAX(hire_date) FROM employees WHERE emp_id = c_emp.emp_id);
BEGIN
 FOR emp IN c_emp LOOP
  DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp.emp_id || ', Name: ' || emp.name || ', Job Title: '
||emp.job_id || ', Start Date: ' || emp.hire_date);
 END LOOP;
END;
```

c)
```
DECLARE
 CURSOR c_emp IS
  SELECT emp_id, salary
  FROM employees
  WHERE dept_id = 50
  FOR UPDATE OF salary;
BEGIN
 FOR emp IN c_emp LOOP
  UPDATE employees
  SET salary = salary * 1.1
  WHERE CURRENT OF c_emp;
 END LOOP;
 COMMIT;
END;
```

d)
```
CREATE OR REPLACE PROCEDURE display_emp_salary_less_than(p_salary_threshold NUMBER) IS
  CURSOR c_emp IS
    SELECT name, salary
    FROM employees
    WHERE salary < p_salary_threshold;
BEGIN
  FOR emp IN c_emp LOOP
    DBMS_OUTPUT.PUT_LINE(emp.name || ': ' || emp.salary);
  END LOOP;
END;
```