Task#1

```
DECLARE
   TYPE namesarray IS
      VARRAY(5) OF VARCHAR2(10);
   TYPE scorearray IS
      VARRAY(5) OF INTEGER;

   studentnames  namesarray;
   studentscores scorearray;
BEGIN
   --initialize data
   studentnames := namesarray('Andy', 'Ram', 'Micheal', 'Sophia', 'Emma');
   studentscores := scorearray(92, 98, 85, 89, 82);


   --loop and output
   FOR i IN 1..studentnames.count LOOP
      dbms_output.put_line('Student: '
                      || studentnames(i)
                      || ', Score = '
                      || studentscores(i));
   END LOOP;

END;
/
```

Output:

```
Student: Andy, Score = 92
Student: Ram, Score = 98
Student: Micheal, Score = 85
Student: Sophia, Score = 89
Student: Emma, Score = 82
```

Task#2

```
DECLARE
   PROCEDURE average_students IS
      TYPE scorearray IS
         VARRAY(5) OF INTEGER;
      studentscores scorearray;
      average       NUMBER;
      total         NUMBER;
   BEGIN
      studentscores := scorearray(92, 98, 85, 89, 82);
      total := 0;
      FOR i IN 1..studentscores.count LOOP
         total := total + studentscores(i);
      END LOOP;

      average := total / studentscores.count;
      dbms_output.put_line('Average is ' || average);
   END;

BEGIN
   average_students;
END;
/
```

Output:

```
Procedure AVERAGE_STUDENTS compiled

Average is 89.2

PL/SQL procedure successfully completed.
```

Task#3

```
DECLARE
   TYPE scorearray IS VARRAY(5) OF INTEGER;

   PROCEDURE display_grades IS
      studentscores scorearray;
      studentname  VARCHAR(20);
      score        NUMBER := 20;
      grade        VARCHAR2(20);
   BEGIN
      studentname := 'Jessica';
      studentscores := scorearray(92, 98, 85, 89, 82);

      FOR i IN 1..studentscores.count LOOP
         CASE
            WHEN studentscores(i) >= 91 AND studentscores(i) <= 100
            THEN
               grade := 'A';
            WHEN studentscores(i) >= 81 AND studentscores(i) <= 90
            THEN
               grade := 'B';
            WHEN studentscores(i) >= 71 AND studentscores(i) <= 80
            THEN
               grade := 'C';
            WHEN studentscores(i) >= 61 AND studentscores(i) <= 70
            THEN
               grade := 'D';
            WHEN studentscores(i) >= 0 AND studentscores(i) <= 60
            THEN
               grade := 'F';
            ELSE
               grade := 'Invalid';
         END CASE;
         dbms_output.put_line('Grade:' || grade);
      END LOOP;
   END;
BEGIN
   display_grades;
END;
/
```

Output:

```
Grade:A
Grade:A
```

```
Grade:B
Grade:B
Grade:B

PL/SQL procedure successfully completed.
```

Task#4

```
DECLARE
   TYPE scorearray IS VARRAY(5) OF INTEGER;
   inputscores scorearray;

   PROCEDURE max_score (studentscores scorearray) IS
      studentname VARCHAR2(20);
      maxscore    NUMBER;
   BEGIN
      -- assume 1st score is the max score
      maxscore := studentscores(1);

      -- perform linear search
      FOR i IN 1..studentscores.count LOOP
         IF studentscores(i) > maxscore THEN
            maxscore := studentscores(i);
         END IF;
      END LOOP;

      -- output
      dbms_output.put_line('Maximum score: ' || maxscore);
   END;

BEGIN
   -- initialize student input scores
   inputscores := scorearray(92, 98, 85, 89, 82);
   -- execute procedure
   max_score(inputscores);
END;
/
```

Output:

```
Maximum score: 98

PL/SQL procedure successfully completed.
```

Task#5

```
CREATE OR REPLACE PACKAGE stu_package AS
   PROCEDURE insert_record (
      id      students.student_id%TYPE,
      name    students.student_name%TYPE,
      age     students.age%TYPE,
      gpa     students.gpa%TYPE,
      address students.address%TYPE
   );

   PROCEDURE delete_record (
      id students.student_id%TYPE
   );

   PROCEDURE update_record (
      id          students.student_id%TYPE,
      new_address students.address%TYPE
   );

   FUNCTION get_average (is_age BOOLEAN) RETURN NUMBER;

   PROCEDURE display_all;

   PROCEDURE display_name_age;

END stu_package;
/

CREATE OR REPLACE PACKAGE BODY stu_package AS

   PROCEDURE insert_record (
      id      students.student_id%TYPE,
      name    students.student_name%TYPE,
      age     students.age%TYPE,
      gpa     students.gpa%TYPE,
      address students.address%TYPE
   ) IS
   BEGIN
      INSERT INTO students VALUES (
         id,
         name,
         age,
         gpa,
        address
    );
    dbms_output.put_line('Student with id ' || id || ' created.');
END insert_record;

PROCEDURE delete_record (
    id students.student_id%TYPE
) IS
BEGIN
    DELETE FROM students
    WHERE
        students.student_id = id;

    dbms_output.put_line('Student with id '|| id || ' deleted.');
END delete_record;

PROCEDURE update_record (
    id          students.student_id%TYPE,
    new_address students.address%TYPE
) IS
BEGIN
    UPDATE students
    SET
        students.address = new_address
    WHERE
        students.student_id = id;

    dbms_output.put_line('Student with id ' || id || ' updated.');
END update_record;

FUNCTION get_average (is_age BOOLEAN) RETURN NUMBER IS
    result NUMBER;
BEGIN
    IF is_age = true THEN
        SELECT
            AVG(age)
        INTO result
        FROM
            students;
    ELSE
        SELECT
            AVG(gpa)
        INTO result
        FROM
```

```
            students;

        END IF;

        RETURN result;
    END;

    PROCEDURE display_all IS

        TYPE stutype IS RECORD (
            student_id      students.student_id%TYPE,
            student_name    students.student_name%TYPE,
            age     students.age%TYPE,
            gpa     students.gpa%TYPE,
            address students.address%TYPE
        );

        stu_obj stutype;

        CURSOR rowtype IS
        SELECT
            student_id,
            student_name,
            age,
            gpa,
            address
        FROM
            students;

    BEGIN
        OPEN rowtype;
        LOOP
            FETCH rowtype INTO stu_obj;
            EXIT WHEN rowtype%notfound;
            dbms_output.put_line(stu_obj.student_id
                                 || '--'
                                 || stu_obj.student_name
                                 || '--'
                                 || stu_obj.age
                                 || '--'
                                 || stu_obj.gpa
                                 || '--'
                                 || stu_obj.address);
```

```
      END LOOP;

      dbms_output.put_line('Number of rows Fetched=' || rowtype%rowcount);
      CLOSE rowtype;
   END display_all;

   PROCEDURE display_name_age IS

      TYPE stutype IS RECORD (
         name    students.student_name%TYPE,
         age     students.age%TYPE
      );
      stu_obj stutype;
      CURSOR rowtype IS
      SELECT student_name, age FROM students;

   BEGIN
      OPEN rowtype;
      LOOP
         FETCH rowtype INTO stu_obj;
         EXIT WHEN rowtype%notfound;
         dbms_output.put_line(stu_obj.name || '--' || stu_obj.age);
      END LOOP;

      dbms_output.put_line('Number of rows Fetched=' || rowtype%rowcount);
      CLOSE rowtype;
   END display_name_age;

END stu_package;
/
```

Output:

```
Package STU_PACKAGE compiled

Package Body STU_PACKAGE compiled
```

Task#6

```
DECLARE
   average NUMBER;
BEGIN
   -- insert records
   stu_package.insert_record(1, 'Andy', 25, 3.6, 'Grand Rapids');
   stu_package.insert_record(2, 'Luke', 30, 3.8, 'Detroit');
   stu_package.insert_record(3, 'Jessi', 28, 3.4, 'Chicago');
   stu_package.insert_record(4, 'Ana', 27, 3.5, 'Colorado');
   stu_package.insert_record(5, 'Sita', 31, 3.7, 'Boston');

   -- delete record
   stu_package.delete_record(1);

   -- updated a record
   stu_package.update_record(1, 'Dallas');

   -- get average
   average := stu_package.get_average(TRUE);
   dbms_output.put_line('Average is ' || average);

   -- display all
   stu_package.display_all();

   -- display name age
   stu_package.display_name_age();

END;
/
```

Output:

```
Student with id 1 created.
Student with id 2 created.
Student with id 3 created.
Student with id 4 created.
Student with id 5 created.
Student with id 1 deleted.
Student with id 1 updated.
Average is 29
2--Luke--30--3.8--Detroit
3--Jessi--28--3.4--Chicago
4--Ana--27--3.5--Colorado
```

```
5--Sita--31--3.7--Boston
Number of rows Fetched=4
Luke--30
Jessi--28
Ana--27
Sita--31
Number of rows Fetched=4

PL/SQL procedure successfully completed.
```