Function

Example#1

Write a PL/SQL function to display welcome message using function.

```
DELIMITER //
CREATE FUNCTION show_message()
RETURNS VARCHAR(100)
NO SQL
BEGIN
   RETURN "welcome to plsql function";
END //
DELIMITER ;

SELECT show_message() AS message;
```

Example#2

Write a PL/SQL function to take a number and find out cube of it.

```
DELIMITER //
CREATE FUNCTION find_cube(x INT)
RETURNS INT
NO SQL
BEGIN
   DECLARE x_cube INT;
   SET x_cube = x * x * x;
          RETURN x_cube;
END //
DELIMITER ;

SELECT find_cube(3);
```

Example#3

Write a function to read age and display eligibility for voting.

```
DELIMITER //
CREATE FUNCTION check_eligibility(p_age INT)
RETURNS VARCHAR(100)
NO SQL
BEGIN
   DECLARE message VARCHAR(100);
   IF p_age >= 18 THEN
      SET message = 'Eligible for voting';
   ELSE
      SET message = 'Not eligible for voting';
   END IF;
   RETURN message;
END //
DELIMITER ;

SELECT check_eligibility(30);
```

Example#4

Write a function to read number and display it is even or odd.

```
DELIMITER //
CREATE FUNCTION check_even_odd (p_number INT)
RETURNS VARCHAR(50)
NO SQL
BEGIN
          DECLARE message VARCHAR(50);
   IF  MOD(p_number, 2) = 0 THEN
      SET message =  CONCAT(p_number, ' is even');
   ELSE
      SET message =  CONCAT(p_number, ' is odd');
   END IF;
   RETURN message;
END //
```

```
DELIMITER ;

SELECT check_even_odd(3);
```

Example#5

Write a function to read 2 numbers and display the greater number.

```
DELIMITER //
CREATE FUNCTION check_greater_number(p_number1 INT, p_number2 INT)
RETURNS INT
NO SQL
BEGIN
          DECLARE greater_number INT;
   IF p_number1 > p_number2 THEN
      SET greater_number =  p_number1;
   ELSE
      SET greater_number =  p_number2;
   END IF;
   RETURN greater_number;
END //
DELIMITER ;

SELECT check_greater_number(30,20);
```

Example#6

Write a PL/SQL stored function for passing employee_id as a parameter, display employee details.

```
DELIMITER //
CREATE FUNCTION get_employee(p_employee_id INT)
RETURNS VARCHAR(200)
READS SQL DATA
BEGIN
   DECLARE v_employee_details VARCHAR(200);
```

```
    SELECT CONCAT(employee_id, employee_name,salary, department_id)
    INTO v_employee_details
    FROM employees
    WHERE employee_id = p_employee_id;

    RETURN v_employee_details;
END //
DELIMITER ;

SELECT get_employee(101);
```

<u>Example#7</u>

Write a PL/SQL stored function for passing employee_id as a parameter, display employee and department details.

```
DELIMITER //

CREATE FUNCTION get_emp_dept(p_employee_id INT)
RETURNS JSON
READS SQL DATA
BEGIN
    DECLARE employee_details JSON;
    SELECT JSON_OBJECT(
        'employee_id', e.employee_id,
        'employee_name', e.employee_name,
        'salary', e.salary,
        'department_id', d.department_id,
        'department_name', d.department_name
    )
    INTO employee_details
    FROM employees e, departments d
    WHERE e.department_id = d.department_id
    AND e.employee_id = p_employee_id;
    RETURN employee_details;
END //
DELIMITER ;

SELECT get_emp_dept(101);
```

Example#8

Write a function get all records.

```
DELIMITER //

CREATE FUNCTION get_emp_dept2()
RETURNS JSON
READS SQL DATA
BEGIN
    DECLARE employee_details JSON;
    SELECT JSON_ARRAYAGG(
        JSON_OBJECT(
            'employee_id', e.employee_id,
            'employee_name', e.employee_name,
            'salary', e.salary,
            'department_id', d.department_id,
            'department_name', d.department_name
        )
    )
    INTO employee_details
    FROM employees e, departments d
    WHERE e.department_id = d.department_id;

    RETURN employee_details;
END //
DELIMITER ;

SELECT get_emp_dept2();
```