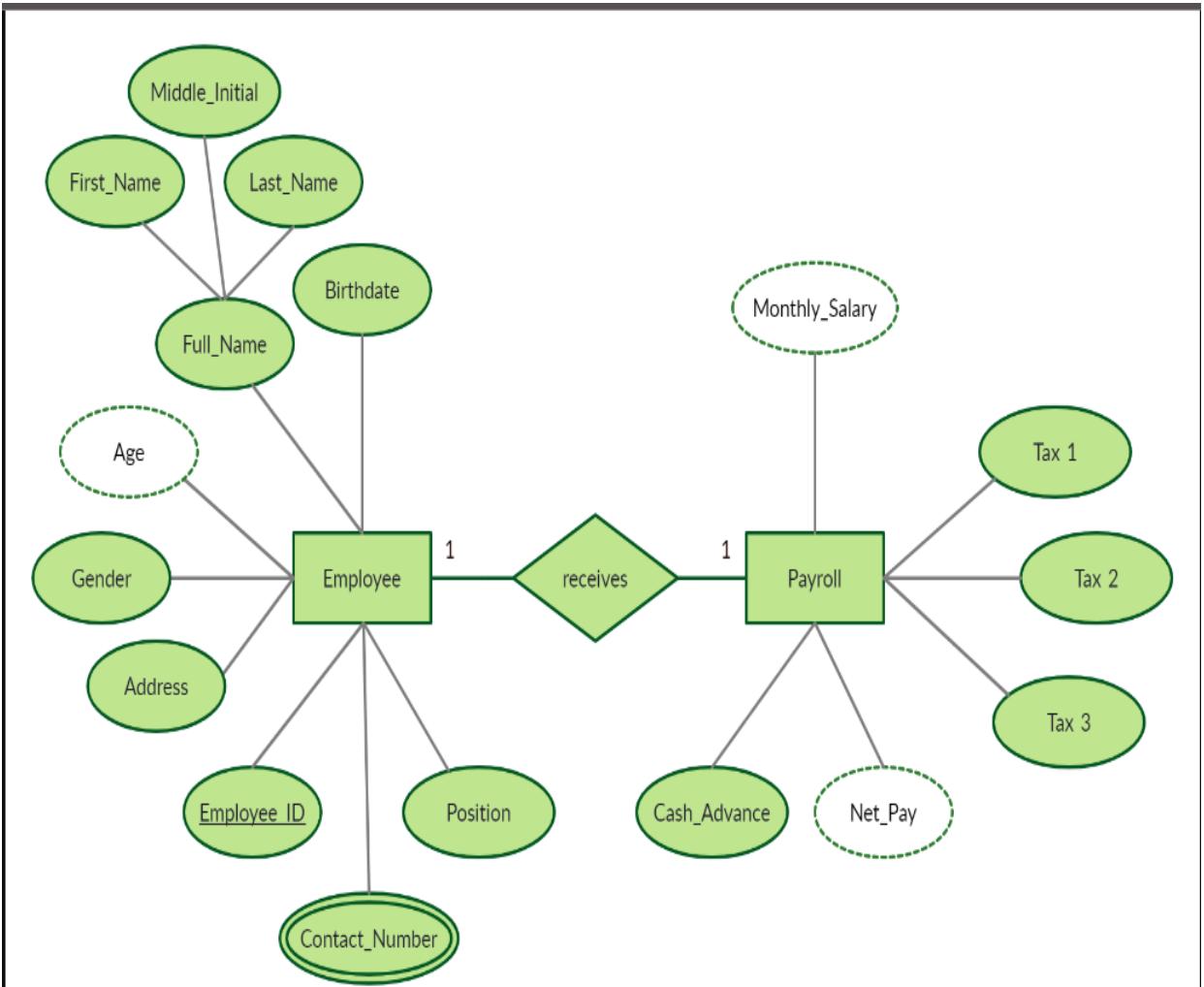


Assignment No 1



Assignment No 2

```
1 • show Databases;
2 • use student;
3
4 • CREATE TABLE Employee (
5     EmployeeID INT PRIMARY KEY,
6     FirstName VARCHAR(50) NOT NULL,
7     LastName VARCHAR(50) NOT NULL,
8     Email VARCHAR(100) UNIQUE,
9     Salary DECIMAL(10, 2)
0 );
1
2 • CREATE TABLE Employee1 (
3     EmployeeID INT PRIMARY KEY,
4     Email VARCHAR(100) UNIQUE,
5     Salary DECIMAL(10, 2)
6 );
7
```

```
17
18 • CREATE TABLE Employee2 (
19     EmployeeID INT PRIMARY KEY,
20     Salary DECIMAL(10, 2) CHECK (Salary >= 30000)
21 );
22
23 • CREATE TABLE Employee3 (
24     EmployeeID INT PRIMARY KEY,
25     Status VARCHAR(20) DEFAULT 'Active'
26 );
27
28 • CREATE TABLE Department (
29     DepartmentID INT PRIMARY KEY,
30     DepartmentName VARCHAR(50)
31 );
```

```

25     Status VARCHAR(20) DEFAULT 'Active'
26 );
27
28 • CREATE TABLE Department (
29     DepartmentID INT PRIMARY KEY,
30     DepartmentName VARCHAR(50)
31 );
32
33 • CREATE TABLE Employee4 (
34     EmployeeID INT PRIMARY KEY,
35     DepartmentID INT REFERENCES Department(DepartmentID)
36 );
37
38 • Show tables;
39

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Result Grid

Tables_in_student
department
employee
employee1
employee2
employee3
employee4
fullname

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	21:38:47	show Databases	6 row(s) returned	0.000 sec / 0.000 sec
2	21:39:09	use student	0 row(s) affected	0.000 sec
3	21:39:20	CREATE TABLE Employee (EmployeeID INT PRIMARY KEY, FirstName VARCHAR(50) NOT NULL, LastName VARCHAR(50) NO...)	0 row(s) affected	0.046 sec
4	21:40:19	CREATE TABLE Employee1 (EmployeeID INT PRIMARY KEY, Email VARCHAR(100) UNIQUE, Salary DECIMAL(10,2))	0 row(s) affected	0.031 sec
5	21:40:31	CREATE TABLE Employee2 (EmployeeID INT PRIMARY KEY, Salary DECIMAL(10,2) CHECK (Salary >= 30000))	0 row(s) affected	0.031 sec
6	21:41:03	CREATE TABLE Employee3 (EmployeeID INT PRIMARY KEY, Status VARCHAR(20) DEFAULT 'Active')	0 row(s) affected	0.031 sec
7	21:41:29	CREATE TABLE Department (DepartmentID INT PRIMARY KEY, DepartmentName VARCHAR(50))	0 row(s) affected	0.031 sec
8	21:41:29	CREATE TABLE Employee4 (EmployeeID INT PRIMARY KEY, DepartmentID INT REFERENCES Department(DepartmentID))	0 row(s) affected	0.016 sec
9	21:41:43	Show tables;	7 row(s) returned	0.000 sec / 0.000 sec

Assignment No 3

Query 1 SQL File 3* X

```

1 • show databases;
2 • use student;
3 • CREATE TABLE Employee5 (
4     EmployeeID INT PRIMARY KEY,
5     FirstName VARCHAR(50) NOT NULL,
6     LastName VARCHAR(50) NOT NULL,
7     Email VARCHAR(100) UNIQUE,
8     Salary DECIMAL(10, 2),
9     DepartmentID INT,
10    HireDate DATE
11 );
12
13

```

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Output

Action Output	#	Time	Action	Message	Duration / Fetch
	9	21:41:43	Show tables	7 row(s) returned	0.000 sec / 0.000 sec
	10	21:54:38	show databases	6 row(s) returned	0.000 sec / 0.000 sec
	11	21:54:51	use student	0 row(s) affected	0.000 sec
	12	21:58:12	INSERT INTO Employee (EmployeeID, FirstName, LastName, DepartmentID, Salary) VALUES (1, 'John', 'Doe', 1, 50000)	Error Code: 1054. Unknown column 'Department...' 0.016 sec	
	13	21:59:14	CREATE TABLE Employee5 (EmployeeID INT PRIMARY KEY, FirstName VARCHAR(50) NOT NULL, LastName VARCHAR(50) ... 0 row(s) affected		0.031 sec

Context Help Snippets

Query 1 SQL File 3* X

```

12 ----- INSERT Statement:
13 ----- Insert a new employee into the "Employee" table.
14 • INSERT INTO Employee5 (EmployeeID, FirstName, LastName, Email, Salary, DepartmentID, HireDate)
15   VALUES (1, 'John', 'Doe', 'john.doe@example.com', 50000, 1, '2023-01-15');
16
17 • select * from Employee5;
18
19

```

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content: Result Grid Form Editor

EmployeeID	FirstName	LastName	Email	Salary	DepartmentID	HireDate
1	John	Doe	john.doe@example.com	50000.00	1	2023-01-15
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Employee5 2 X

Output

Action Output	#	Time	Action	Message	Duration / Fetch
	12	21:58:12	INSERT INTO Employee (EmployeeID, FirstName, LastName, DepartmentID, Salary) VALUES (1, 'John', 'Doe', 1, 50000)	Error Code: 1054. Unknown column 'Department...' 0.016 sec	
	13	21:59:14	CREATE TABLE Employee5 (EmployeeID INT PRIMARY KEY, FirstName VARCHAR(50) NOT NULL, LastName VARCHAR(50) ... 0 row(s) affected		0.031 sec
	14	22:00:50	INSERT INTO Employee (EmployeeID, FirstName, LastName, DepartmentID, Salary) VALUES (1, 'John', 'Doe', 1, 50000)	Error Code: 1054. Unknown column 'Department...' 0.000 sec	
	15	22:01:54	INSERT INTO Employee5 (EmployeeID, FirstName, LastName, Email, Salary, DepartmentID, HireDate) VALUES (1, 'John', 'Doe', 'john.doe@example.com', 50000, 1, '2023-01-15')	1 row(s) affected	0.015 sec
	16	22:02:15	select * from Employee5 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Context Help Snippets

Query 1 SQL File 3*

```

19     VALUES (2, 'Jane', 'Smith', 'jane.smith@example.com', 60000, 4, '2023-02-20');
20
21 • select * from Employee5;
22
23 • INSERT INTO Department (DepartmentID, DepartmentName)
24   VALUES (2, 'Marketing');
25
26 • INSERT INTO Department (DepartmentID, DepartmentName)
27   VALUES (3, 'Finance');
28
29 • INSERT INTO Department (DepartmentID, DepartmentName)
30   VALUES (4, 'Sales');
31
32 ----- UPDATE Statement:
33 ----- Increase the salary of employees in the "Sales" department by 10%.
34 • UPDATE Employee5
35   SET Salary = Salary * 1.10

```

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content: Result Grid

EmployeeID	FirstName	LastName	Email	Salary	DepartmentID	HireDate
1	John	Doe	john.doe@example.com	50000.00	1	2023-01-15
2	Jane	Smith	jane.smith@example.com	60000.00	4	2023-02-20
*	NULL	NULL	NULL	NULL	NULL	NULL

Query 1 SQL File 3*

```

33 ----- Increase the salary of employees in the "Sales" department by 10%.
34 • UPDATE Employee5
35   SET Salary = Salary * 1.10
36   WHERE DepartmentID=4;
37 • select * from Employee5;
38
39 • DELETE FROM Employee5
40   WHERE Salary < 30000;
41
42 • select * from Employee5;

```

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content: Result Grid

EmployeeID	FirstName	LastName	Email	Salary	DepartmentID	HireDate
1	John	Doe	john.doe@example.com	50000.00	1	2023-01-15
2	Jane	Smith	jane.smith@example.com	60000.00	4	2023-02-20
*	NULL	NULL	NULL	NULL	NULL	NULL

Employee5 4* Context Help

Action Output

#	Time	Action	Message
1	22:29:01	DELETE FROM Employee5 WHERE Salary < 30000	0 row(s) affected
2	22:29:05	select * from Employee5 LIMIT 0, 1000	2 row(s) returned

41
42 • SELECT EmployeeID, FirstName, LastName, Salary
43 FROM Employee5
44 WHERE Salary > (SELECT AVG(Salary) FROM Employee);
45
46 • select * from Employee5;

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content: Result Grid

EmployeeID	FirstName	LastName	Salary
*	NULL	NULL	NULL

Employee5 7* Context Help

Assignment 4:

Query 1 SQL File 3* | Limit to 1000 rows | Context Help

```
46 • select * from Employee5;
47
48 • CREATE TABLE Employee9 (
49     EmployeeID INT PRIMARY KEY,
50     FirstName VARCHAR(50) NOT NULL,
51     LastName VARCHAR(50) NOT NULL,
52     Email VARCHAR(100) UNIQUE,
53     Salary DECIMAL(10, 2),
54     DepartmentID INT
55 );
56 • CREATE VIEW EmployeeSummary AS
57     SELECT EmployeeID, FirstName, LastName, Salary
58     FROM Employee;
59
60 • CREATE INDEX EmailIndex
61     ON Employee (Email);
62
63
```

Output Action Output

#	Time	Action	Message
1	22:34:15	CREATE VIEW EmployeeSummary AS SELECT EmployeeID...	0 row(s) affected
2	22:34:29	CREATE INDEX EmailIndex ON Employee (Email)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

Query 1 SQL File 3* | Limit to 1000 rows

```
CREATE TABLE Employee9 (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    Salary DECIMAL(10, 2),
    DepartmentID INT
);
CREATE VIEW EmployeeSummary AS
SELECT EmployeeID, FirstName, LastName, Salary
FROM Employee;

CREATE INDEX EmailIndex
ON Employee (Email);

CREATE SEQUENCE EmployeeIDSequence
START WITH 1
INCREMENT BY 1;

CREATE SYNONYM EmployeeInfo FOR Employee;
```

Assignment No 5:

The screenshot shows two separate result grids from SQL Server Management Studio. The top grid displays the results of a query that inserts sample sales data into the Sales table and then counts the total number of rows. The bottom grid displays the results of a query that calculates the total sales amount by summing the quantity times price for all rows in the Sales table.

Result Grid 1 (Top):

TotalOrders
3

Result Grid 2 (Bottom):

TotalSales
333.77

Query 1 SQL File 3* ×

```
85     VALUES (3, 'Widget', 8, 12.99);
86
87 •   SELECT COUNT(*) AS TotalOrders
88     FROM Sales;
89
90 •   SELECT SUM(Quantity * Price) AS TotalSales
91     FROM Sales;
92
93 •   SELECT MAX(Price) AS MaxPrice
94     FROM Sales;
95
96 •   SELECT MIN(Price) AS MinPrice
97     FROM Sales;
98
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Result Grid

MaxPrice
19.99

Query 1 SQL File 3* ×

```
91     FROM Sales;
92
93 •   SELECT MAX(Price) AS MaxPrice
94     FROM Sales;
95
96 •   SELECT MIN(Price) AS MinPrice
97     FROM Sales;
98
99 •   SELECT AVG(Price) AS AvgPrice
100    FROM Sales;
101
102
103
104
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Result Grid

MinPrice
12.99

Result 11 × Read Only

Assignment No 6:

```
108
109      -- Create the Orders table
110 •   CREATE TABLE Orders (
111         OrderID INT PRIMARY KEY,
112         CustomerID INT,
113         OrderDate DATE,
114         TotalAmount DECIMAL(10, 2),
115         FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
116     );
117
118      -- Insert sample data into the Customers table
119 •   INSERT INTO Customers (CustomerID, FirstName, LastName)
120     VALUES (1, 'John', 'Doe');
121
122 •   INSERT INTO Customers (CustomerID, FirstName, LastName)
123     VALUES (2, 'Jane', 'Smith');
124
125      -- Insert sample data into the Orders table
126 •   INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)
127     VALUES (101, 1, '2023-10-15', 100.50);
128
129 •   INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)
130     VALUES (102, 1, '2023-10-16', 75.20);
131
```

The screenshot shows a database management interface with several panes:

- Top pane:** Displays the SQL code for creating the Orders table and inserting sample data into the Customers and Orders tables.
- Second pane:** Shows the execution of an INSERT query for an order (OrderID 104) and a SELECT query joining the Orders and Customers tables.
- Third pane:** A Result Grid showing the data from the SELECT query, with four rows: (101, John, Doe), (102, John, Doe), (103, Jane, Smith), and (104, Jane, Smith).
- Bottom pane:** An Action Output log showing the history of actions taken, including the insertion of four rows into the Orders table and the execution of the SELECT query.

```

141
142
143 •  SELECT c.CustomerID, c.FirstName, c.LastName, o.OrderID
144   FROM Customers c
145   LEFT JOIN Orders o ON c.CustomerID = o.CustomerID;
146
147

```

Result Grid | Filter Rows: _____ | Export: | Wrap Cell Content:

CustomerID	FirstName	LastName	OrderID
1	John	Doe	101
1	John	Doe	102
2	Jane	Smith	103
2	Jane	Smith	104

Result 13 x

Output

Action Output			
#	Time	Action	Message
16	23:12:29	INSERT INTO Orders (OrderID, CustomerID, OrderDate, T...	1 row(s) affected
17	23:12:33	INSERT INTO Orders (OrderID, CustomerID, OrderDate, T...	1 row(s) affected
18	23:12:36	INSERT INTO Orders (OrderID, CustomerID, OrderDate, T...	1 row(s) affected
19	23:12:40	SELECT o.OrderID, c.FirstName, c.LastName FROM Order...	4 row(s) returned
20	23:13:13	SELECT c.CustomerID, c.FirstName, c.LastName, o.Orderi...	4 row(s) returned

Query 1 SQL File 3* x

FROM Customers c
LEFT JOIN Orders o ON c.CustomerID = o.CustomerID;

144
145
146
147 • SELECT c1.FirstName, c1.LastName, c2.FirstName, c2.LastName
148 FROM Customers c1
149 INNER JOIN Customers c2 ON c1.LastName = c2.LastName AND c1.CustomerID <> c2.CustomerID;
150
151
152
153

Result Grid | Filter Rows: _____ | Export: | Wrap Cell Content:

FirstName	LastName	FirstName	LastName
John	Doe	Jane	Smith

Result 14 x

Output

Action Output			
#	Time	Action	Message
17	23:12:33	INSERT INTO Orders (OrderID, CustomerID, OrderDate, T...	1 row(s) affected
18	23:12:36	INSERT INTO Orders (OrderID, CustomerID, OrderDate, T...	1 row(s) affected
19	23:12:40	SELECT o.OrderID, c.FirstName, c.LastName FROM Order...	4 row(s) returned
20	23:13:13	SELECT c.CustomerID, c.FirstName, c.LastName, o.Orderi...	4 row(s) returned
21	23:14:32	SELECT c1.FirstName, c1.LastName, c2.FirstName, c2.La...	0 row(s) returned

Query 1 SQL File 3* x

```
148     FROM Customers c1
149     INNER JOIN Customers c2 ON c1.LastName = c2.LastName AND c1.CustomerID <> c2.CustomerID;
150
151 •   SELECT FirstName, LastName
152     FROM Customers
153     WHERE CustomerID IN (
154         SELECT CustomerID
155         FROM Orders
156         WHERE TotalAmount > 150.00
157     );
```

Result Grid | Filter Rows: Export: Wrap Cell Content: □

FirstName	LastName
Jane	Smith

Customers 15 x

Output Read Only

Query 1 SQL File 3* x

```
160     FROM Customers c
161     WHERE EXISTS (
162         SELECT 1
163         FROM Orders o
164         WHERE o.CustomerID = c.CustomerID
165         AND o.TotalAmount > (
166             SELECT AVG(TotalAmount)
167             FROM Orders
168             WHERE CustomerID = c.CustomerID
169         )
170     );
```

Result Grid | Filter Rows: Export: Wrap Cell Content: □

FirstName	LastName
John	Doe
Jane	Smith

Form Editor

Assignment No 7:

```
-- Declare variables to hold the minimum number of students and their scores.  
DECLARE  
    v_min_students NUMBER := 10;  
    v_student_id NUMBER;  
    v_student_score NUMBER;  
    v_grade VARCHAR2(1);  
BEGIN  
    -- Loop through the students with the highest scores (TOP 10).  
    FOR student_rec IN (  
        SELECT StudentID, Score  
        FROM StudentScores  
        WHERE ROWNUM <= v_min_students  
        ORDER BY Score DESC  
    ) LOOP  
        v_student_id := student_rec.StudentID;  
        v_student_score := student_rec.Score;  
  
        -- Calculate the grade based on the student's score.  
        IF v_student_score >= 90 THEN  
            v_grade := 'A';  
        ELSIF v_student_score >= 80 THEN  
            v_grade := 'B';  
        ELSIF v_student_score >= 70 THEN  
            v_grade := 'C';  
        ELSIF v_student_score >= 60 THEN  
            v_grade := 'D';  
        ELSE  
            v_grade := 'F';  
        END IF;  
  
        -- Display the result.  
        DBMS_OUTPUT.PUT_LINE('Student ' || v_student_id || ' scored ' || v_student_score || ', Grade: ' || v_grade);  
    END LOOP;  
END;
```

Assignment No 8:

```
DECLARE
    -- Implicit Cursor (SQL%ROWCOUNT and SQL%NOTFOUND are implicit cursor attributes)
    v_count NUMBER;

    -- Explicit Cursor
    CURSOR employee_cursor IS
        SELECT EmployeeID, FirstName, LastName FROM Employees;
    v_emp_id NUMBER;
    v_emp_firstname VARCHAR2(50);
    v_emp_lastname VARCHAR2(50);

    -- Cursor Variable (Dynamic SQL)
    TYPE ref_cursor IS REF CURSOR;
    v_cursor_variable ref_cursor;

BEGIN
    -- Implicit Cursor (SELECT statement)
    SELECT COUNT(*) INTO v_count FROM Employees;

    -- Display result
    DBMS_OUTPUT.PUT_LINE('Total Employees: ' || v_count);

    -- Explicit Cursor (Fetch and display employee data)
    OPEN employee_cursor;
    LOOP
        -- Display result
        DBMS_OUTPUT.PUT_LINE('Total Employees: ' || v_count);

        -- Explicit Cursor (Fetch and display employee data)
        OPEN employee_cursor;
        LOOP
            FETCH employee_cursor INTO v_emp_id, v_emp_firstname, v_emp_lastname;
            EXIT WHEN employee_cursor%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_emp_id || ', Name: ' || v_emp_firstname || ' ' || v_emp_lastname);
        END LOOP;
        CLOSE employee_cursor;

        -- Cursor Variable (Dynamic SQL)
        OPEN v_cursor_variable FOR
            'SELECT DepartmentName FROM Departments WHERE DepartmentID = :dept_id'
        USING 1; -- Bind variable value
        FETCH v_cursor_variable INTO v_emp_firstname;
        CLOSE v_cursor_variable;

        -- Display result
        DBMS_OUTPUT.PUT_LINE('Department Name for DepartmentID 1: ' || v_emp_firstname);
    END;
    /

```

Assignment No 9:

```
--> ----- procedure
| CREATE OR REPLACE PROCEDURE CalculateDepartmentSalary(
|   p_department_id NUMBER,
|   p_total_salary OUT NUMBER
| )
| IS
| BEGIN
|   SELECT SUM(Salary)
|   INTO p_total_salary
|   FROM Employee
|   WHERE DepartmentID = p_department_id;
|
|   DBMS_OUTPUT.PUT_LINE('Total salary for department ' || p_department_id || ': ' || p_total_salary);
| END;
| /
```



```
--> ----- Function
| CREATE OR REPLACE FUNCTION CalculateBonus(
|   p_salary NUMBER
| ) RETURN NUMBER
| IS
|   v_bonus NUMBER;
| BEGIN
|   IF p_salary >= 50000 THEN
|     v_bonus := p_salary * 0.1; -- 10% bonus for high earners
|   ELSE
|     v_bonus := p_salary * 0.05; -- 5% bonus for others
|   END IF;
|
|   RETURN v_bonus;
| END;
| /
```

Assignment No 10:

```
-----Row-Level Database Trigger:  
CREATE OR REPLACE TRIGGER UpdateLastModified  
BEFORE INSERT OR UPDATE ON YourTable  
FOR EACH ROW  
BEGIN  
    :NEW.LastModified := SYSTIMESTAMP;  
END;  
----- Statement-Level Database Trigger:  
  
CREATE OR REPLACE TRIGGER LogDeleteOperation  
AFTER DELETE ON YourTable  
BEGIN  
    INSERT INTO DeleteLog (OperationTime, TableName, RowsDeleted)  
    VALUES (SYSTIMESTAMP, 'YourTable', SQL%ROWCOUNT);  
END;  
/
```