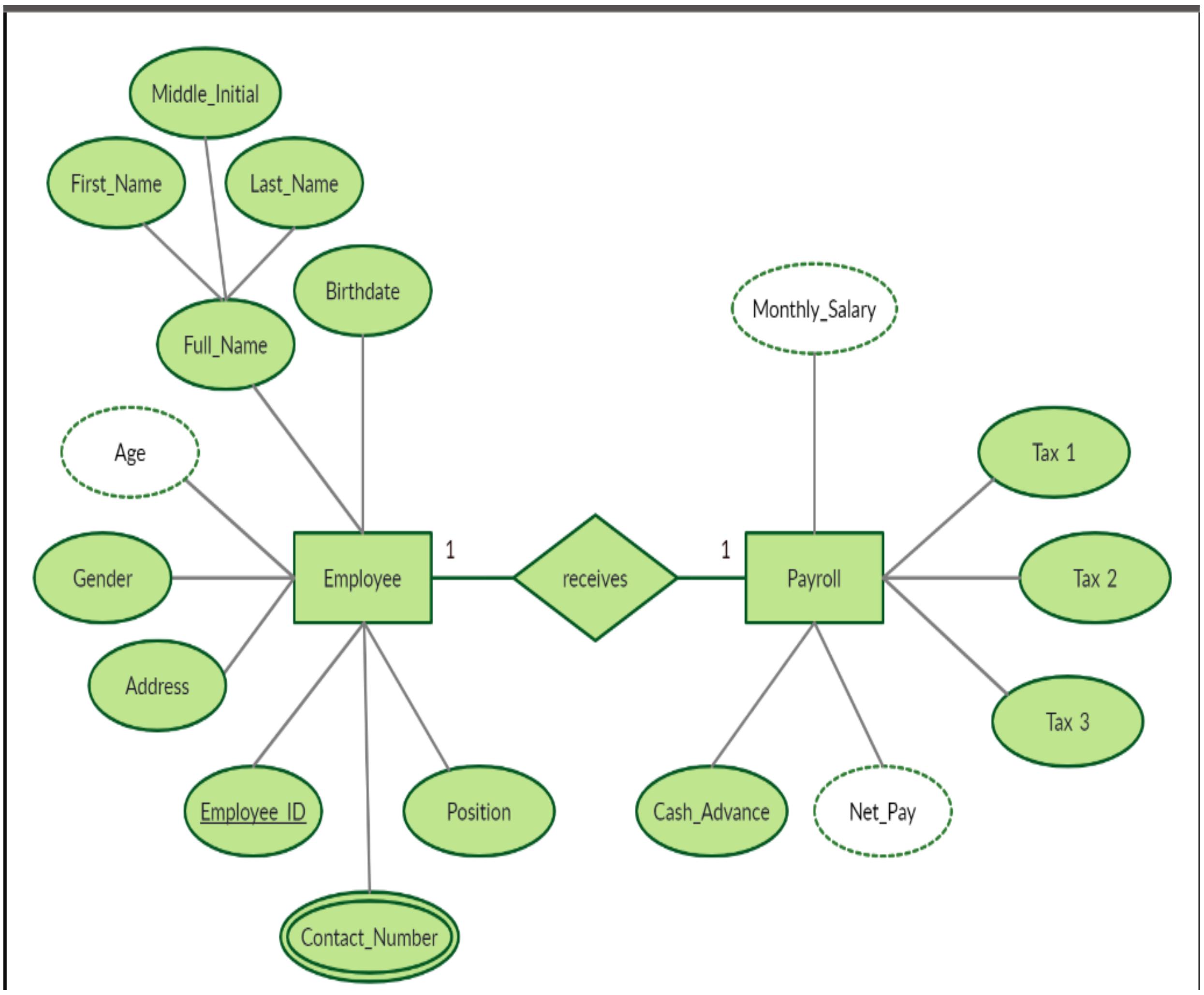
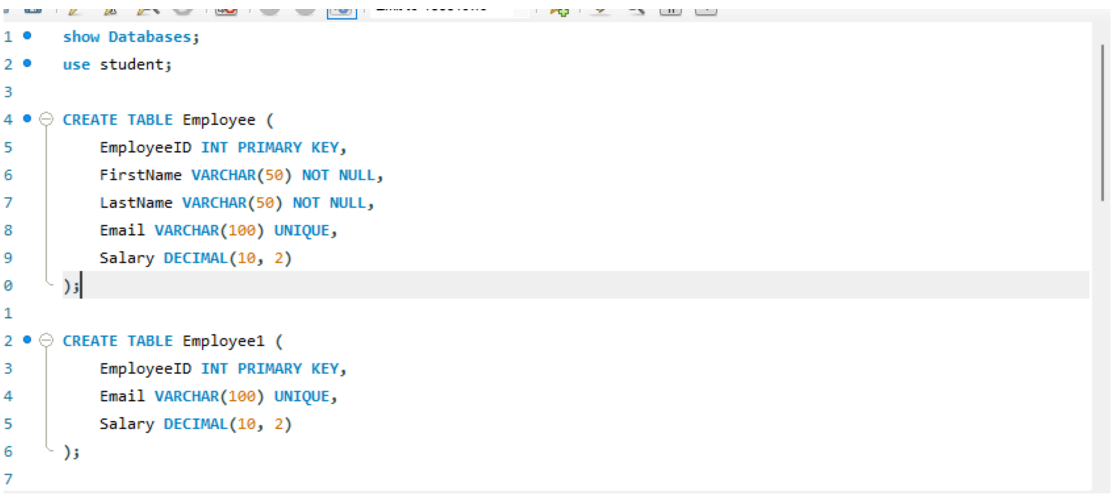
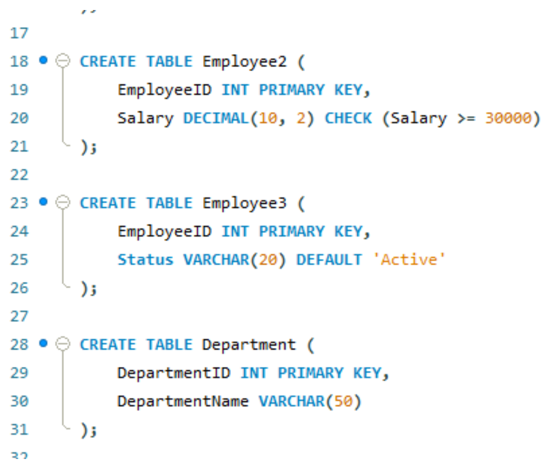
Assignment No 1



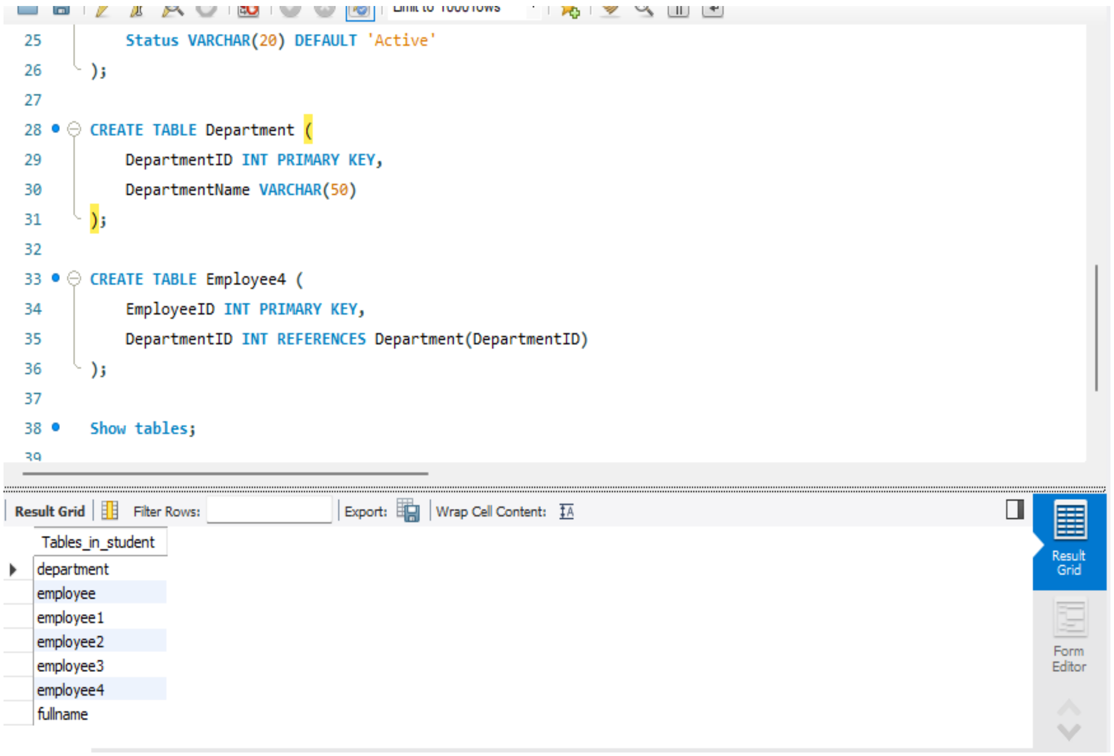
Assignment No 2



```sql
show Databases;
use student;

CREATE TABLE Employee (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    Salary DECIMAL(10, 2)
);

CREATE TABLE Employee1 (
    EmployeeID INT PRIMARY KEY,
    Email VARCHAR(100) UNIQUE,
    Salary DECIMAL(10, 2)
);
```

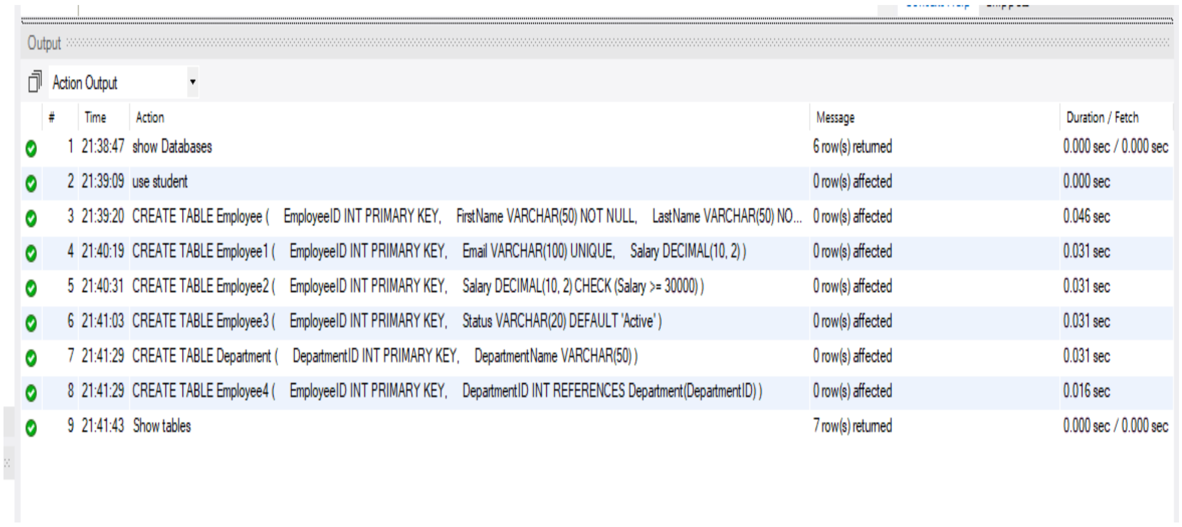


```sql
CREATE TABLE Employee2 (
    EmployeeID INT PRIMARY KEY,
    Salary DECIMAL(10, 2) CHECK (Salary >= 30000)
);

CREATE TABLE Employee3 (
    EmployeeID INT PRIMARY KEY,
    Status VARCHAR(20) DEFAULT 'Active'
);

CREATE TABLE Department (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(50)
);
```



```sql
    Status VARCHAR(20) DEFAULT 'Active'
);

CREATE TABLE Department (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(50)
);

CREATE TABLE Employee4 (
    EmployeeID INT PRIMARY KEY,
    DepartmentID INT REFERENCES Department(DepartmentID)
);

Show tables;
```

Tables_in_student
department
employee
employee1
employee2
employee3
employee4
fullname

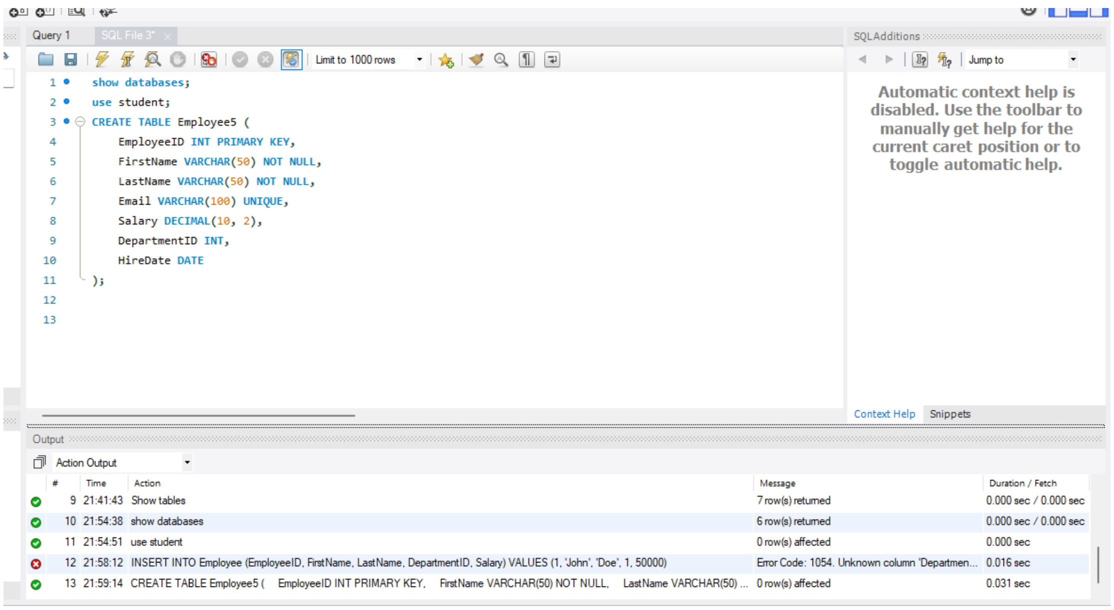


Output

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

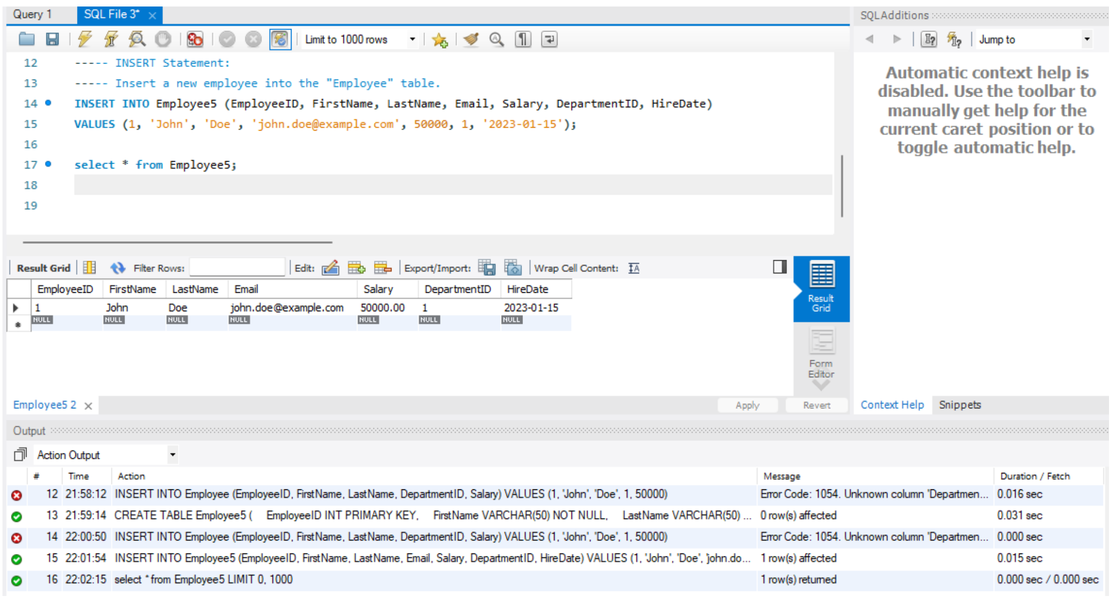
Assignment No 3



```sql
show databases;
use student;
CREATE TABLE Employee5 (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    Salary DECIMAL(10, 2),
    DepartmentID INT,
    HireDate DATE
);
```

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

```
#   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 'Departmen...   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
```

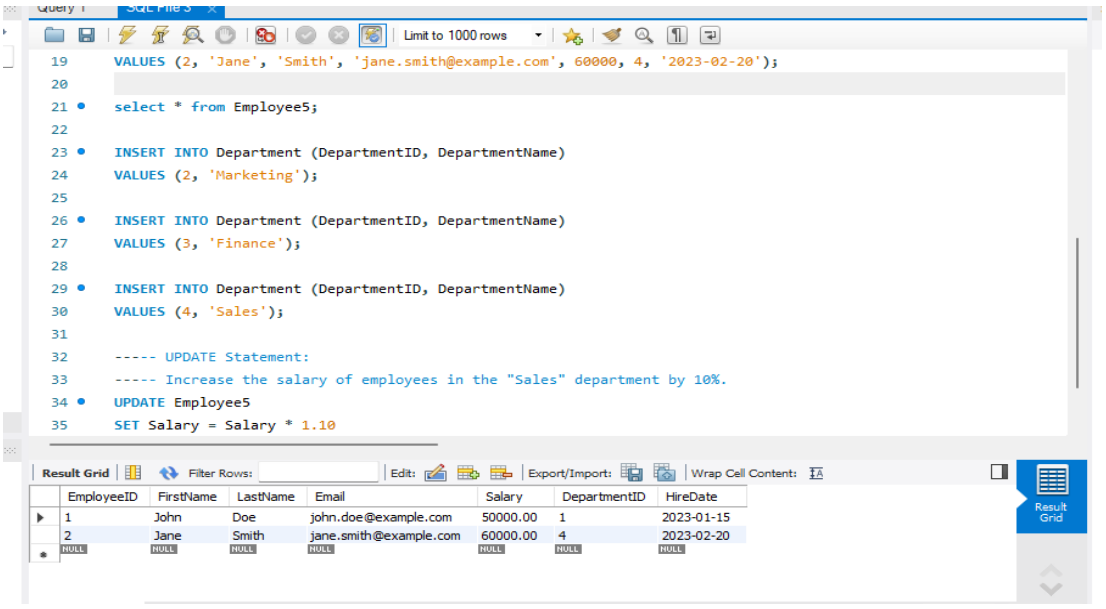


```sql
----- INSERT Statement:
----- Insert a new employee into the "Employee" table.
INSERT INTO Employee5 (EmployeeID, FirstName, LastName, Email, Salary, DepartmentID, HireDate)
VALUES (1, 'John', 'Doe', 'john.doe@example.com', 50000, 1, '2023-01-15');

select * from Employee5;
```

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

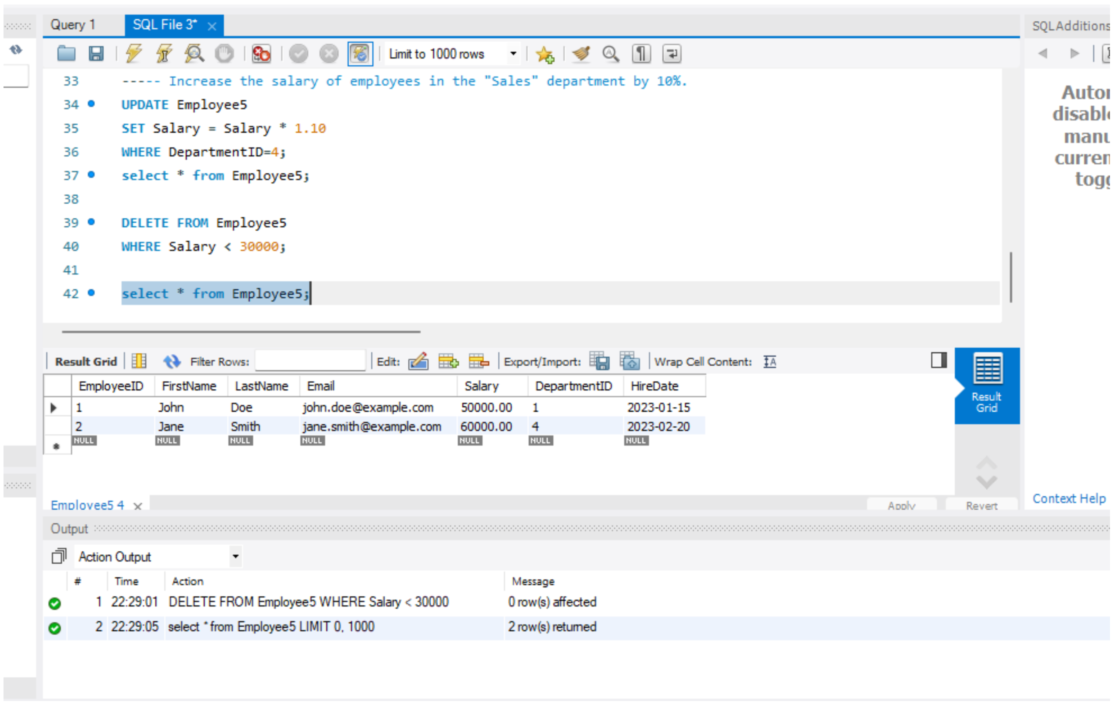
EmployeeID	FirstName	LastName	Email	Salary	DepartmentID	HireDate
1	John	Doe	john.doe@example.com	50000.00	1	2023-01-15
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
#   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 'Departmen...   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 'Departmen...   0.000 sec
15  22:01:54  INSERT INTO Employee5 (EmployeeID, FirstName, LastName, Email, Salary, DepartmentID, HireDate) VALUES (1, 'John', 'Doe', 'john.do...   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
```



```sql
VALUES (2, 'Jane', 'Smith', 'jane.smith@example.com', 60000, 4, '2023-02-20');

select * from Employee5;

INSERT INTO Department (DepartmentID, DepartmentName)
VALUES (2, 'Marketing');

INSERT INTO Department (DepartmentID, DepartmentName)
VALUES (3, 'Finance');

INSERT INTO Department (DepartmentID, DepartmentName)
VALUES (4, 'Sales');

----- UPDATE Statement:
----- Increase the salary of employees in the "Sales" department by 10%.
UPDATE Employee5
SET Salary = Salary * 1.10
```

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	NULL

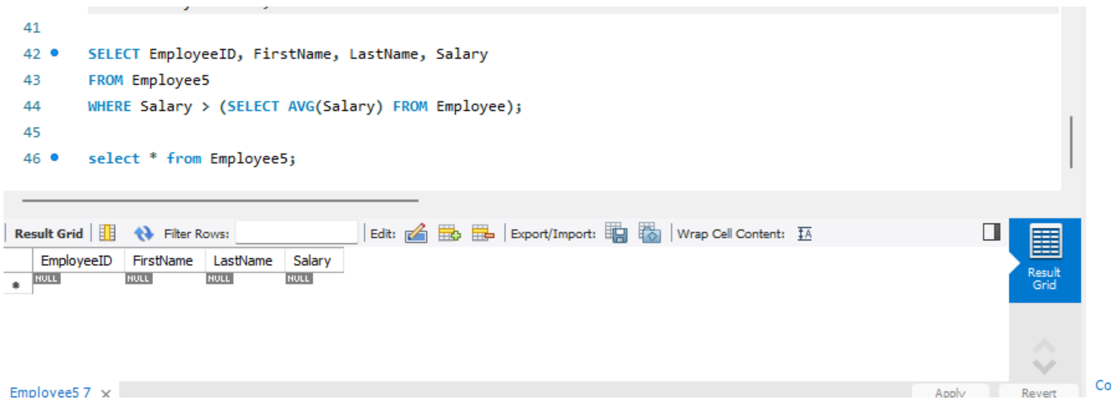


```sql
----- Increase the salary of employees in the "Sales" department by 10%.
UPDATE Employee5
SET Salary = Salary * 1.10
WHERE DepartmentID=4;
select * from Employee5;

DELETE FROM Employee5
WHERE Salary < 30000;

select * from Employee5;
```

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	NULL

Output — Action Output

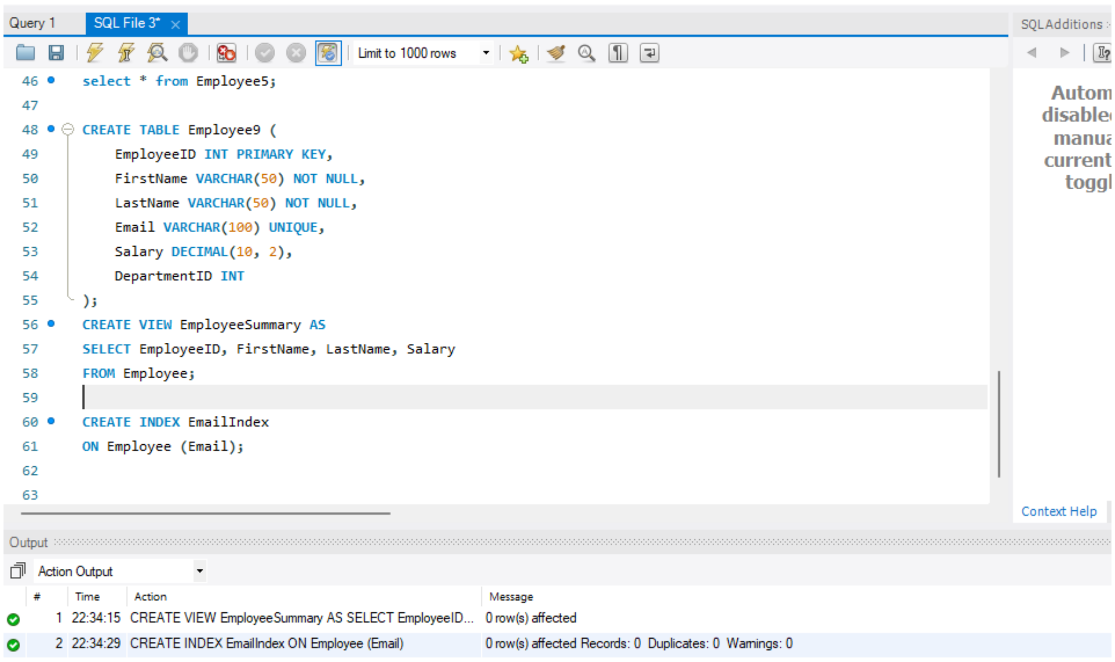
#	Time	Action	Message
1	22:29:01	DELETE FROM Employee5 WHERE Salary < 30000	0 row(s) affected
2	22:29:05	select * from Employee5 LIMIT 0, 1000	2 row(s) returned



```sql
SELECT EmployeeID, FirstName, LastName, Salary
FROM Employee5
WHERE Salary > (SELECT AVG(Salary) FROM Employee);

select * from Employee5;
```

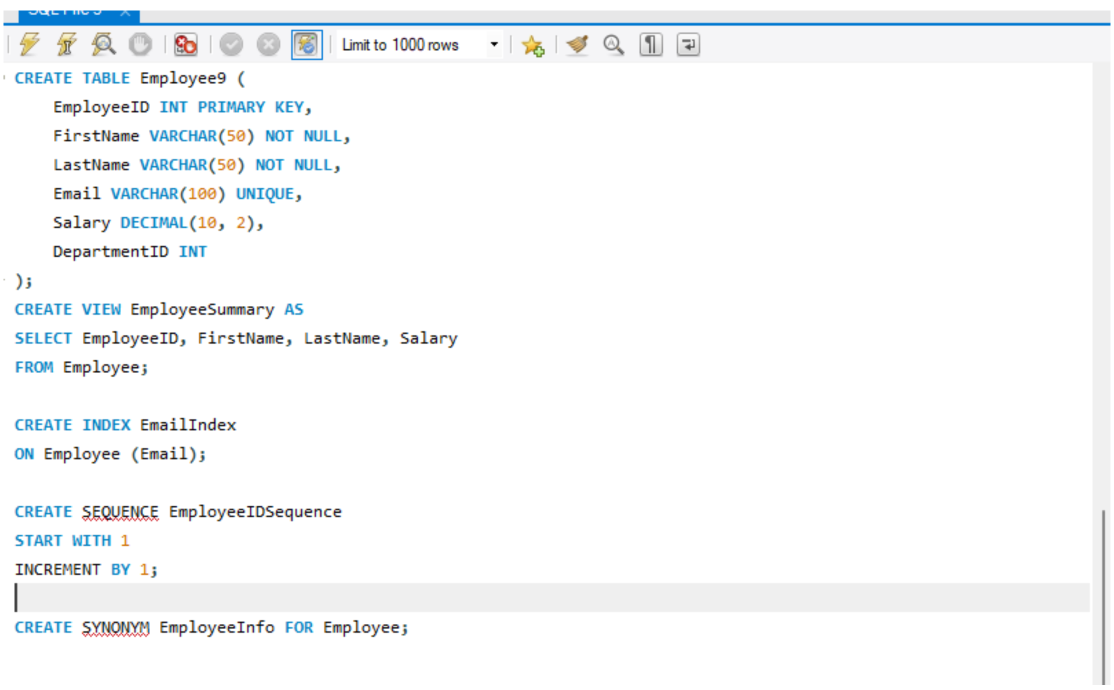
EmployeeID	FirstName	LastName	Salary
NULL	NULL	NULL	NULL

Assignment 4:



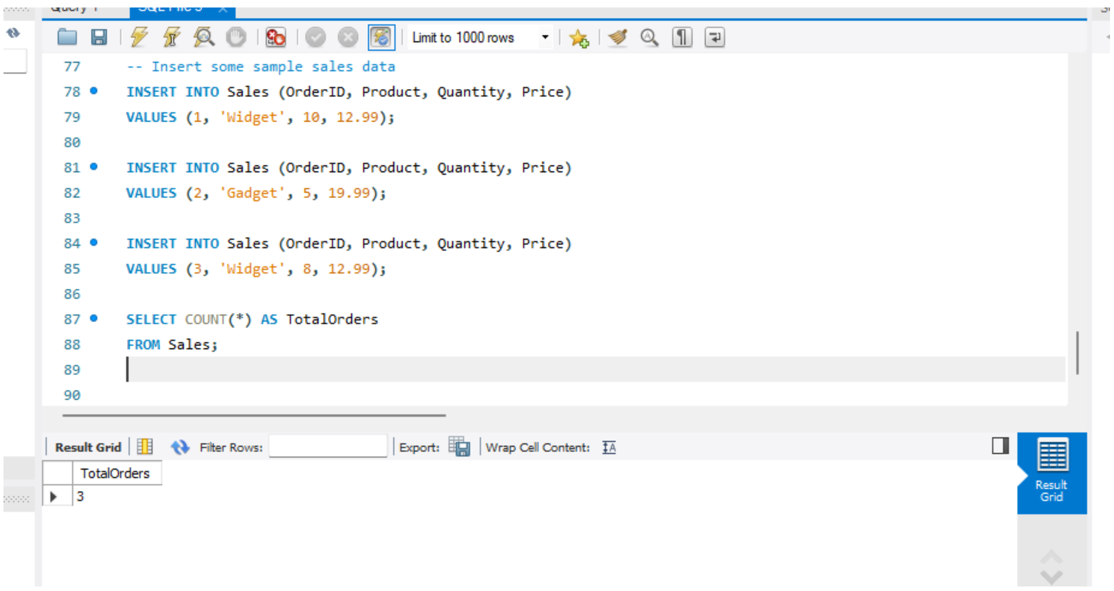
```
select * from Employee5;

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);
```

```
#  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
```



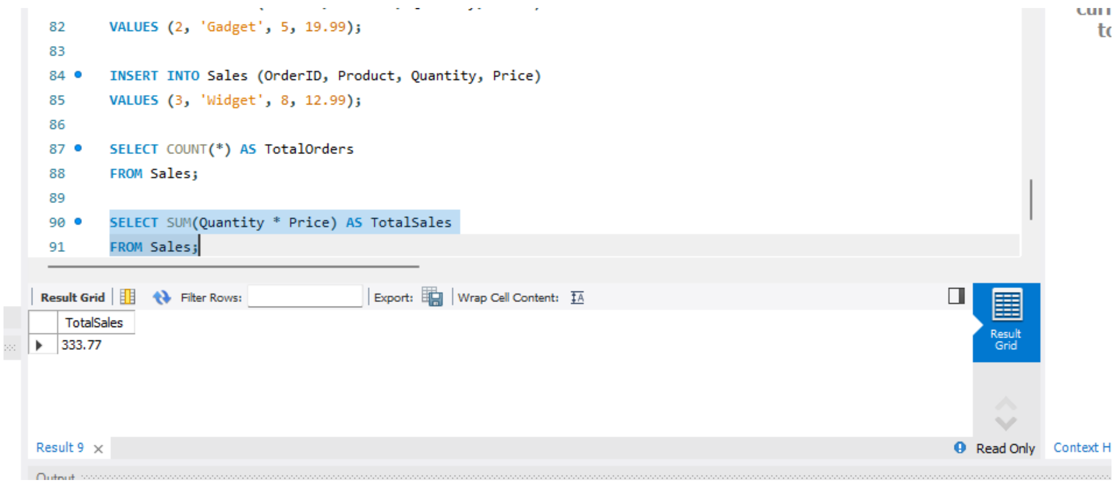
```
CREATE TABLE Employee9 (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    Salary DECIMAL(10, 2),
    DepartmentID INT
);
CREATE VIEW EmployeeSummary AS
SELECT EmployeeID, FirstName, LastName, Salary
FROM Employee;

CREATE INDEX EmailIndex
ON Employee (Email);

CREATE SEQUENCE EmployeeIDSequence
START WITH 1
INCREMENT BY 1;

CREATE SYNONYM EmployeeInfo FOR Employee;
```

Assignment No 5:



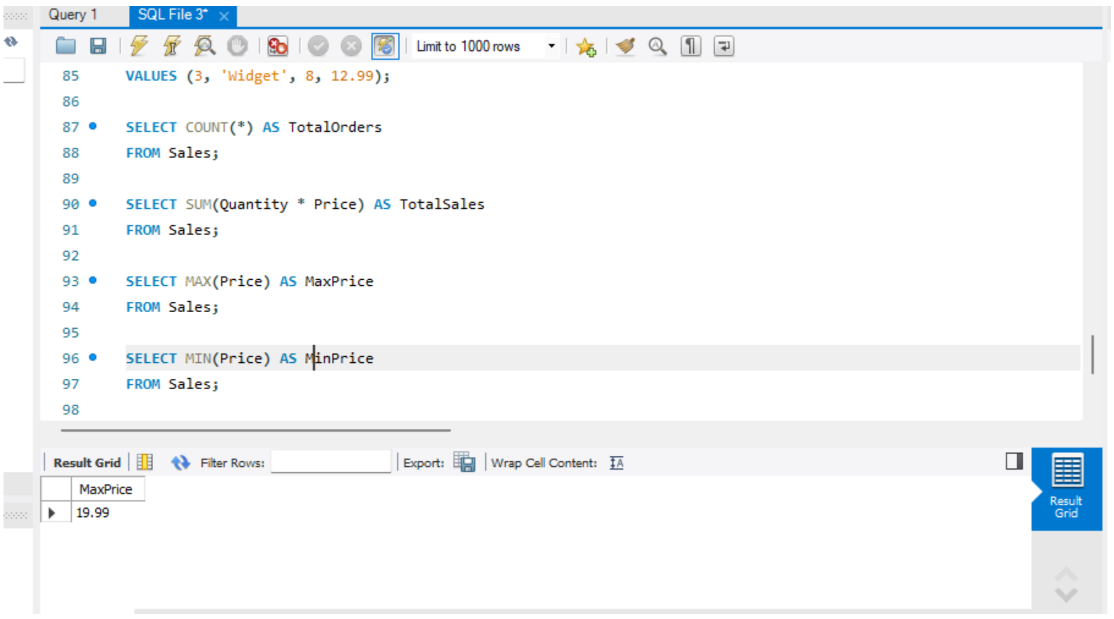
```
-- Insert some sample sales data
INSERT INTO Sales (OrderID, Product, Quantity, Price)
VALUES (1, 'Widget', 10, 12.99);

INSERT INTO Sales (OrderID, Product, Quantity, Price)
VALUES (2, 'Gadget', 5, 19.99);

INSERT INTO Sales (OrderID, Product, Quantity, Price)
VALUES (3, 'Widget', 8, 12.99);

SELECT COUNT(*) AS TotalOrders
FROM Sales;
```

TotalOrders
3



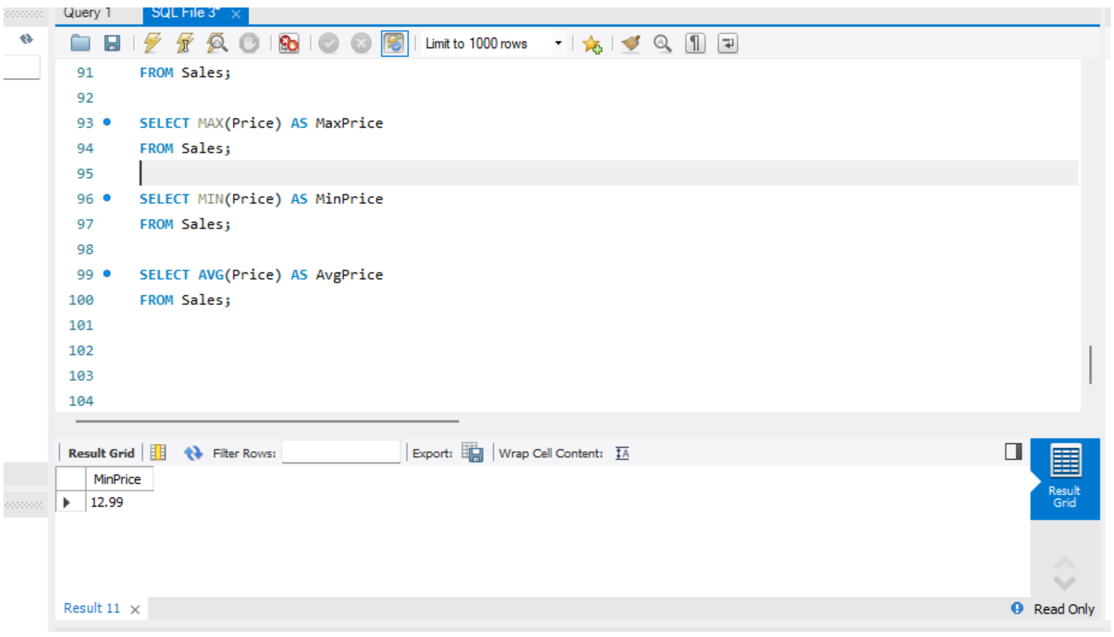
```
VALUES (2, 'Gadget', 5, 19.99);

INSERT INTO Sales (OrderID, Product, Quantity, Price)
VALUES (3, 'Widget', 8, 12.99);

SELECT COUNT(*) AS TotalOrders
FROM Sales;

SELECT SUM(Quantity * Price) AS TotalSales
FROM Sales;
```

TotalSales
333.77



```sql
VALUES (3, 'Widget', 8, 12.99);

SELECT COUNT(*) AS TotalOrders
FROM Sales;

SELECT SUM(Quantity * Price) AS TotalSales
FROM Sales;

SELECT MAX(Price) AS MaxPrice
FROM Sales;

SELECT MIN(Price) AS MinPrice
FROM Sales;
```

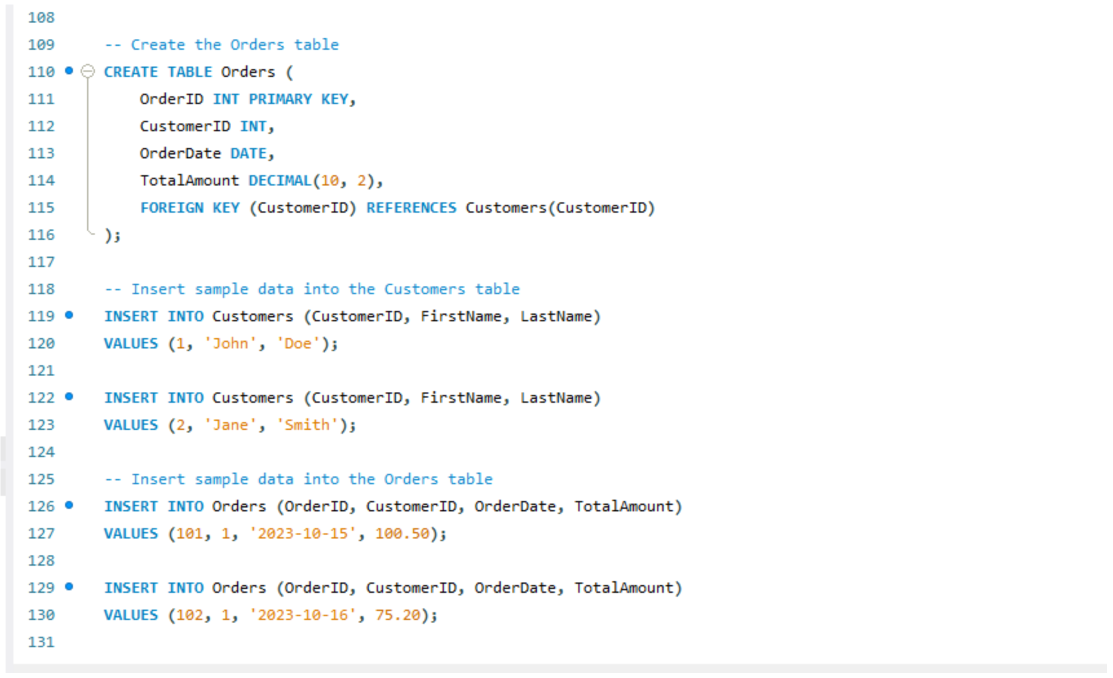
```
MaxPrice
19.99
```



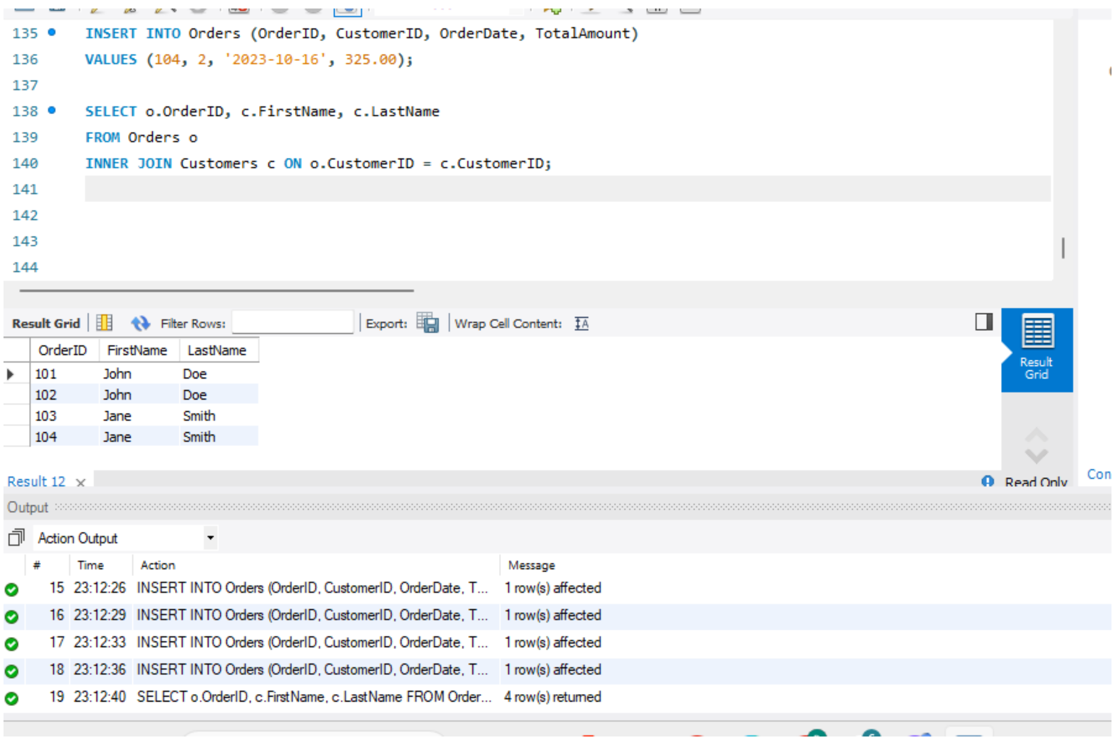
```sql
FROM Sales;

SELECT MAX(Price) AS MaxPrice
FROM Sales;

SELECT MIN(Price) AS MinPrice
FROM Sales;

SELECT AVG(Price) AS AvgPrice
FROM Sales;
```

```
MinPrice
12.99
```

Assignment No 6:



```sql

-- Create the Orders table
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    OrderDate DATE,
    TotalAmount DECIMAL(10, 2),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

-- Insert sample data into the Customers table
INSERT INTO Customers (CustomerID, FirstName, LastName)
VALUES (1, 'John', 'Doe');

INSERT INTO Customers (CustomerID, FirstName, LastName)
VALUES (2, 'Jane', 'Smith');

-- Insert sample data into the Orders table
INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)
VALUES (101, 1, '2023-10-15', 100.50);

INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)
VALUES (102, 1, '2023-10-16', 75.20);

```



```sql
INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)
VALUES (104, 2, '2023-10-16', 325.00);

SELECT o.OrderID, c.FirstName, c.LastName
FROM Orders o
INNER JOIN Customers c ON o.CustomerID = c.CustomerID;
```

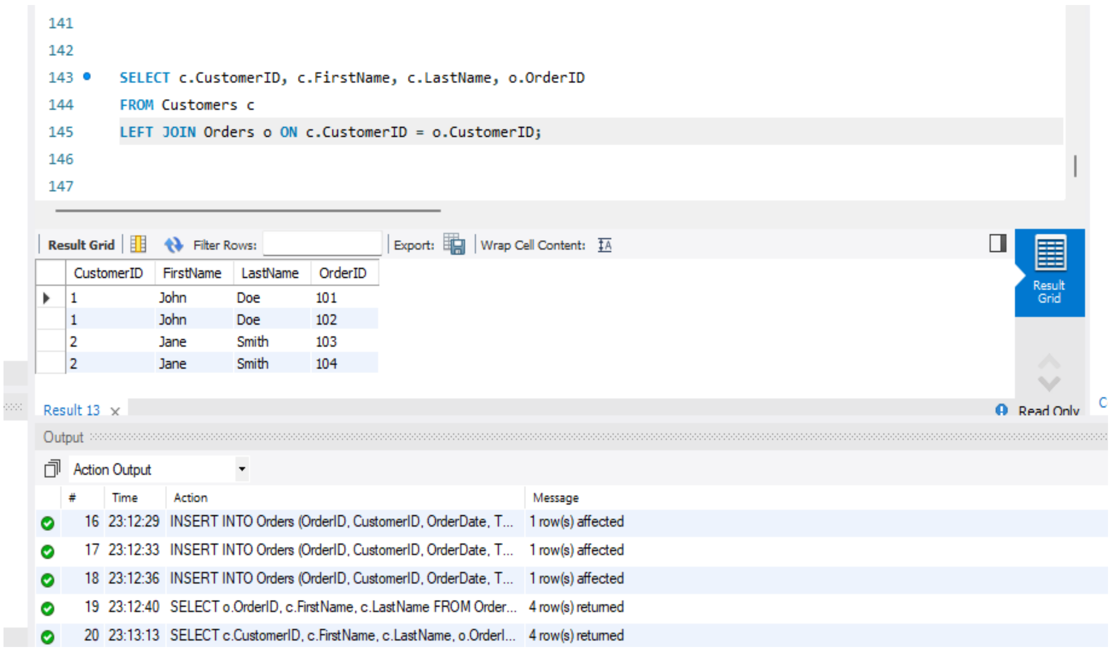
OrderID	FirstName	LastName
101	John	Doe
102	John	Doe
103	Jane	Smith
104	Jane	Smith

Result 12

Output

Action Output

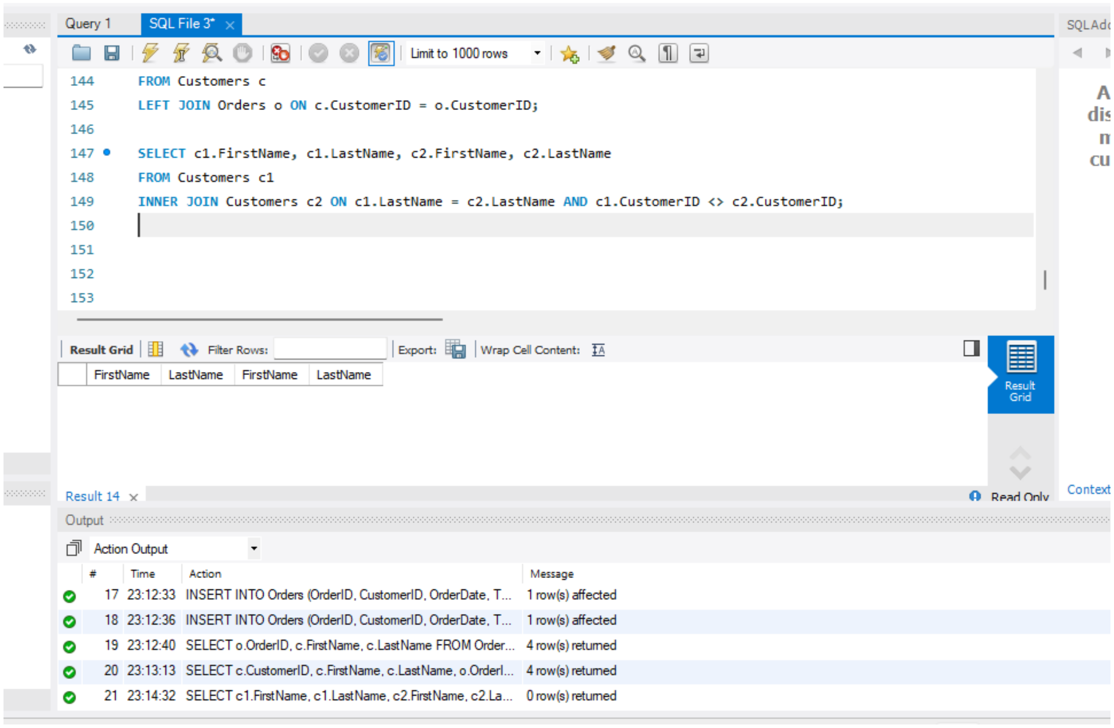
#	Time	Action	Message
15	23:12:26	INSERT INTO Orders (OrderID, CustomerID, OrderDate, T...	1 row(s) affected
16	23:12:29	INSERT INTO Orders (OrderID, CustomerID, OrderDate, T...	1 row(s) affected
17	23:12:33	INSERT INTO Orders (OrderID, CustomerID, OrderDate, T...	1 row(s) affected
18	23:12:36	INSERT 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



```sql
SELECT c.CustomerID, c.FirstName, c.LastName, o.OrderID
FROM Customers c
LEFT JOIN Orders o ON c.CustomerID = o.CustomerID;
```

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

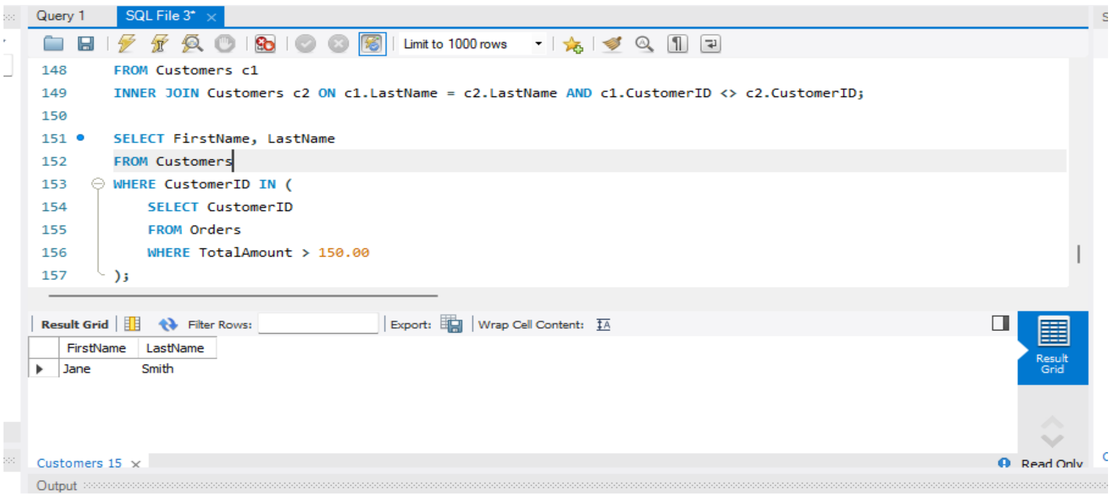
```
#   Time      Action                                               Message
16  23:12:29  INSERT INTO Orders (OrderID, CustomerID, OrderDate, T...  1 row(s) affected
17  23:12:33  INSERT INTO Orders (OrderID, CustomerID, OrderDate, T...  1 row(s) affected
18  23:12:36  INSERT INTO Orders (OrderID, 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
```



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

SELECT c1.FirstName, c1.LastName, c2.FirstName, c2.LastName
FROM Customers c1
INNER JOIN Customers c2 ON c1.LastName = c2.LastName AND c1.CustomerID <> c2.CustomerID;
```

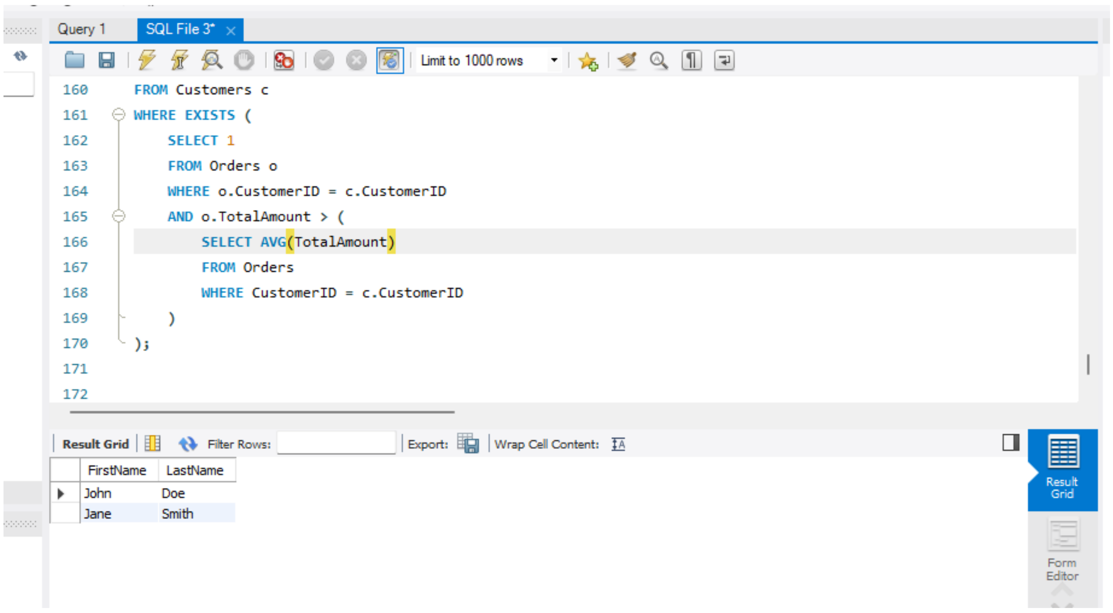
FirstName	LastName	FirstName	LastName

```
#   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
```



```sql
FROM Customers c1
INNER JOIN Customers c2 ON c1.LastName = c2.LastName AND c1.CustomerID <> c2.CustomerID;

SELECT FirstName, LastName
FROM Customers
WHERE CustomerID IN (
    SELECT CustomerID
    FROM Orders
    WHERE TotalAmount > 150.00
);
```

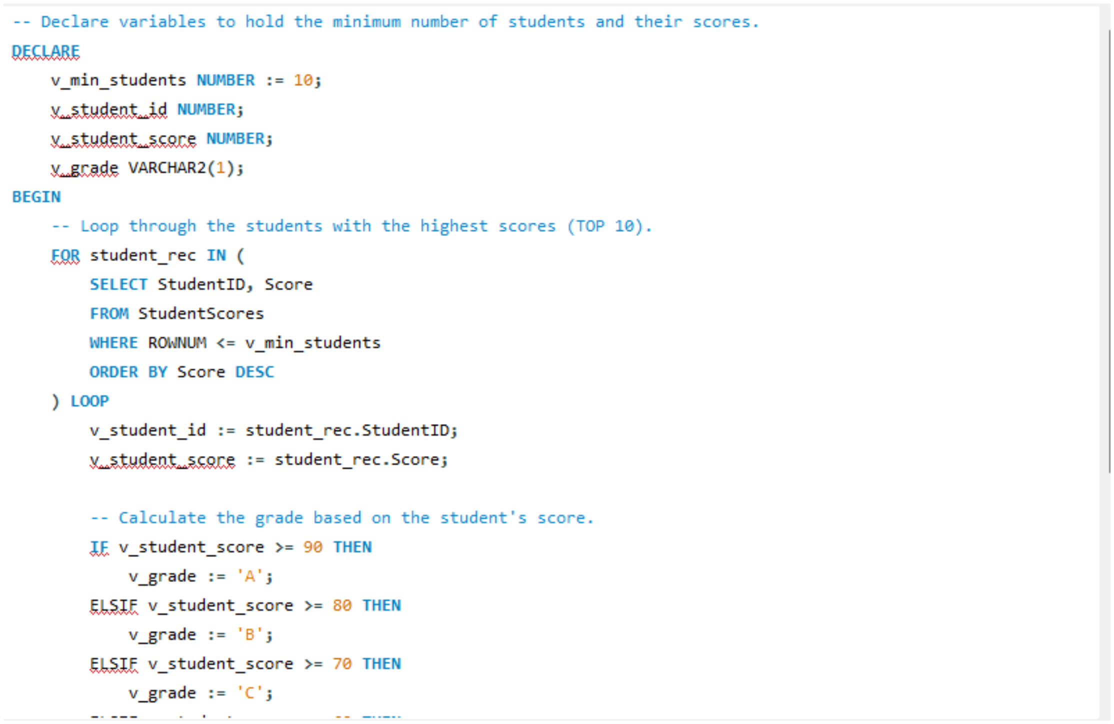
FirstName	LastName
Jane	Smith



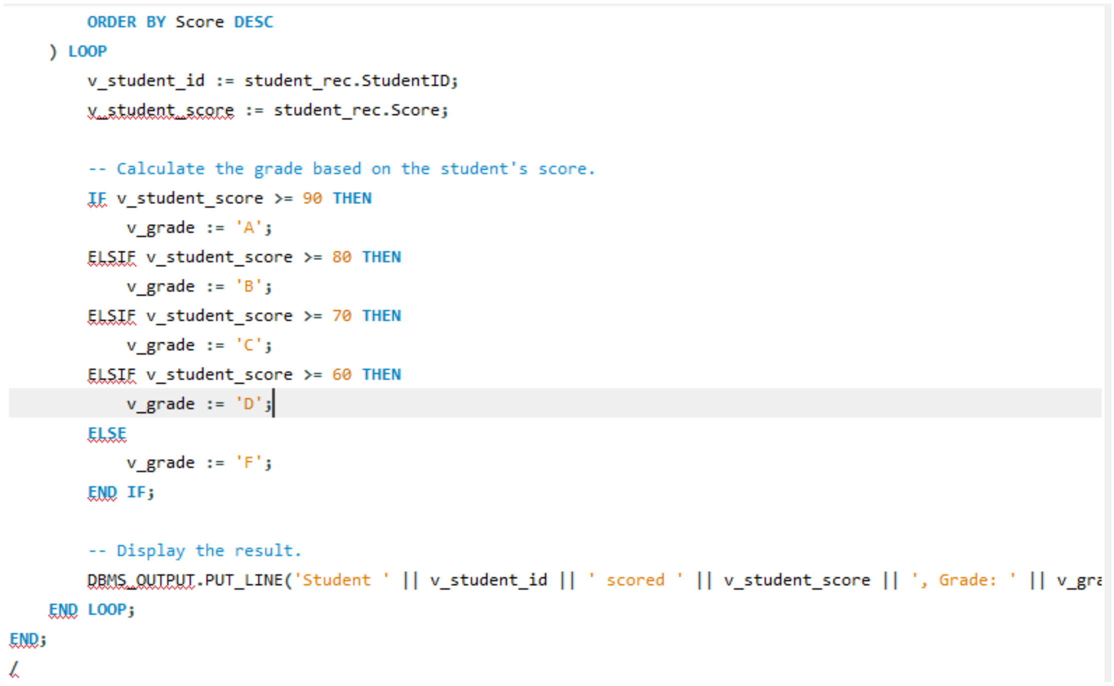
```sql
FROM Customers c
WHERE EXISTS (
    SELECT 1
    FROM Orders o
    WHERE o.CustomerID = c.CustomerID
    AND o.TotalAmount > (
        SELECT AVG(TotalAmount)
        FROM Orders
        WHERE CustomerID = c.CustomerID
    )
);
```

FirstName	LastName
John	Doe
Jane	Smith

Assignment No 7:



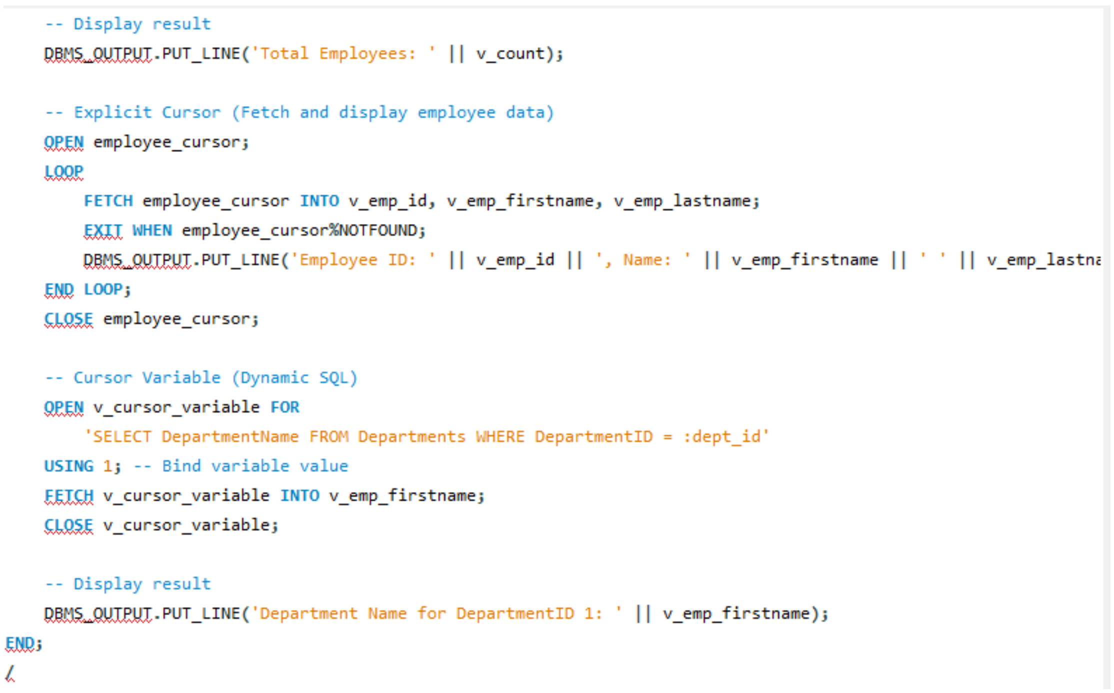
```
-- Declare variables to hold the minimum number of students and their scores.
DECLARE
    v_min_students NUMBER := 10;
    v_student_id NUMBER;
    v_student_score NUMBER;
    v_grade VARCHAR2(1);
BEGIN
    -- Loop through the students with the highest scores (TOP 10).
    FOR student_rec IN (
        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';
```



```
        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_gra
    END LOOP;
END;
/
```

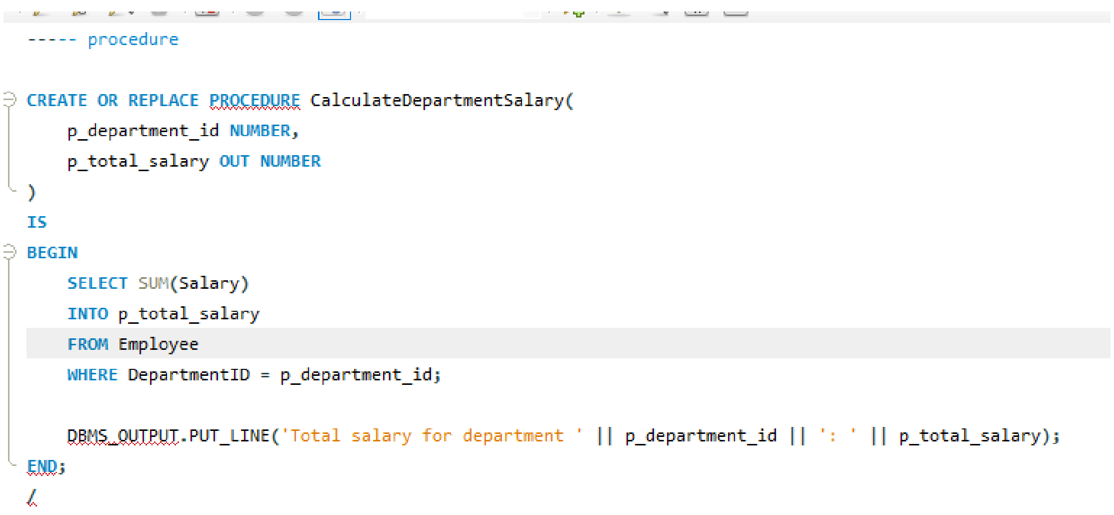
Assignment No 8:

```
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
```

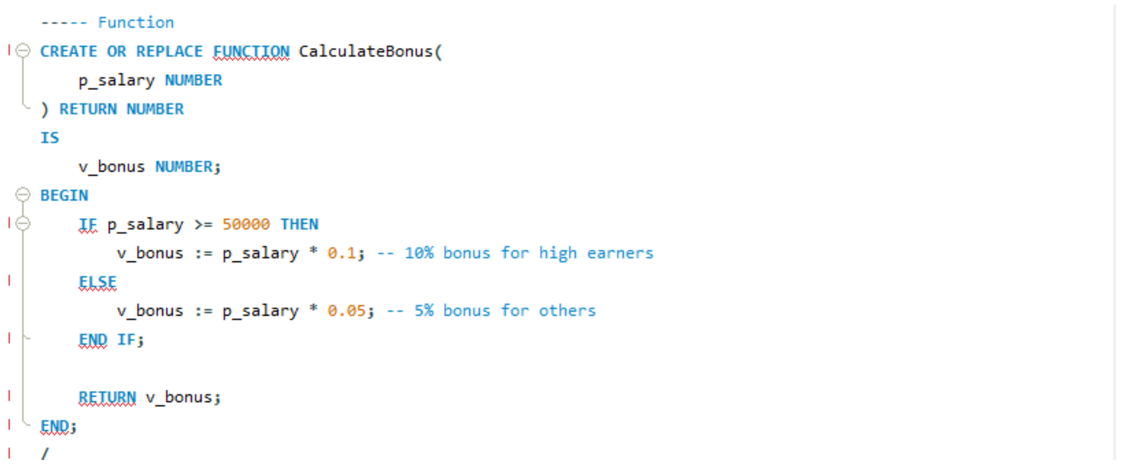


```
    -- Display result
    DBMS_OUTPUT.PUT_LINE('Total Employees: ' || v_count);

    -- 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_lastna
    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:

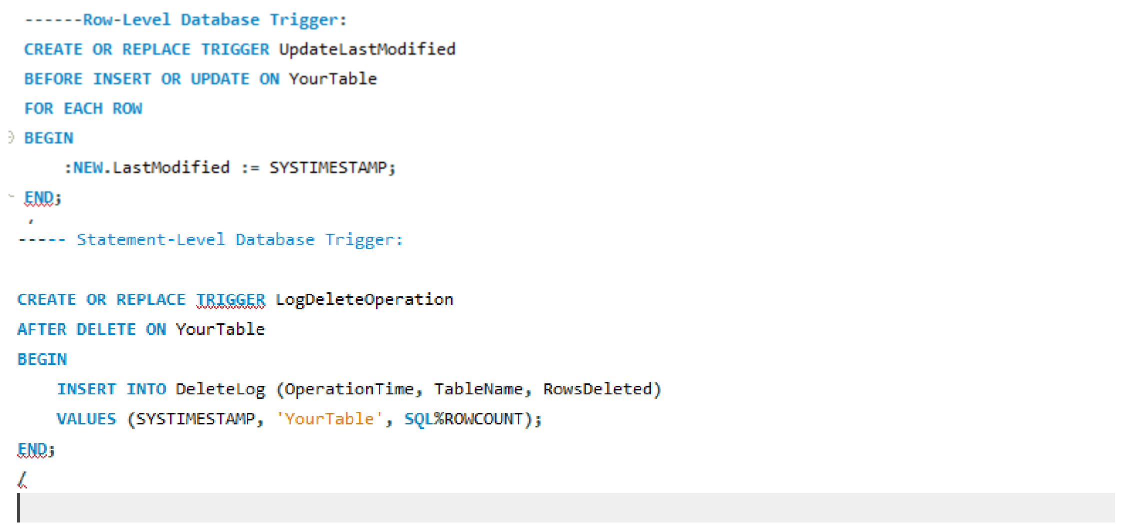


```
----- procedure

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;
/
```