SQL Activity 9

Stored Procedures:

Consider the Worker table with following fields: Worker_Id INT FirstName CHAR (25), LastName CHAR (25), Salary INT, JoiningDate DATETIME, Department CHAR (25))

1. Create a stored procedure that takes in IN parameters for all the columns in the Worker table and adds a new record to the table and then invokes the procedure call.

2. Write stored procedure takes in an IN parameter for WORKER_ID and an OUT parameter for SALARY. It should retrieve the salary of the worker with the given ID and returns it in the p_salary parameter. Then make the procedure call.

3. Create a stored procedure that takes in IN parameters for WORKER_ID and DEPARTMENT. It should update the department of the worker with the given ID. Then make a procedure call.

4. Write a stored procedure that takes in an IN parameter for DEPARTMENT and an OUT parameter for p_workerCount. It should retrieve the number of workers in the given department and returns it in the p_workerCount parameter. Make procedure call.

5. Write a stored procedure that takes in an IN parameter for DEPARTMENT and an OUT parameter for p_avgSalary. It should retrieve the average salary of all workers in the given department and returns it in the p_avgSalary parameter and call the procedure.

->In this document, we will outline a series of stored procedures for managing worker data in an industry. These procedures include inserting new worker records, retrieving worker salaries, updating worker departments, counting workers in a department, and calculating average salaries in a department. We will also demonstrate how to call these procedures with example data relevant to an industry.

-> creating table Worker.

```
CREATE TABLE Worker(
 Worker_Id INT PRIMARY KEY,
 FirstName CHAR(25),
 LastName CHAR(25),
 Salary INT,
 JoiningDate DATETIME,
 Department CHAR(25));
```

->Inserting data in country table.

INSERT INTO Worker (Worker_Id, FirstName, LastName, Salary, JoiningDate, Department) VALUES

(1, 'Raj', 'Kumar', 75000, '2023-05-01 08:00:00', 'Drilling'),

(2, 'Priya', 'Sharma', 82000, '2023-06-15 08:00:00', 'Geology'),

(3, 'Vikram', 'Singh', 68000, '2024-01-10 09:00:00', 'Engineering'),

(4, 'Anjali', 'Verma', 75000, '2023-03-20 10:00:00', 'Drilling'),

(5, 'Amit', 'Patel', 72000, '2022-11-25 11:00:00', 'Maintenance'),

(6, 'Sneha', 'Gupta', 69000, '2023-08-05 08:30:00', 'Exploration'),

(7, 'Arjun', 'Reddy', 76000, '2023-07-18 09:30:00', 'Engineering'),

(8, 'Ravi', 'Joshi', 73000, '2023-09-12 10:00:00', 'Drilling'),

(9, 'Lakshmi', 'Menon', 81000, '2024-02-05 11:00:00', 'Geology'),

(10, 'Kiran', 'Desai', 78000, '2023-04-22 08:45:00', 'Exploration'),

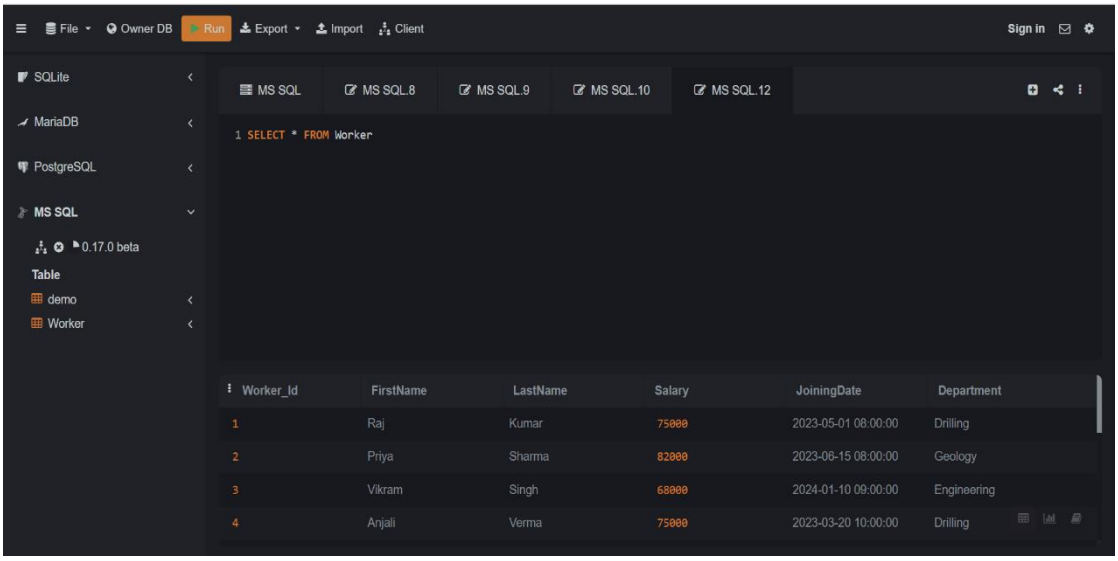
(11, 'Pooja', 'Rao', 69000, '2023-10-10 09:15:00', 'Maintenance'),

(12, 'Suresh', 'Nair', 77000, '2023-12-01 10:30:00', 'Engineering'),

(13, 'Neha', 'Mehta', 80000, '2024-01-25 11:15:00', 'Geology'),

(14, 'Rohit', 'Khan', 70000, '2023-06-10 08:00:00', 'Drilling'),

(15, 'Meera', 'Iyer', 72000, '2023-03-30 09:45:00', 'Exploration');



Store Procedure:

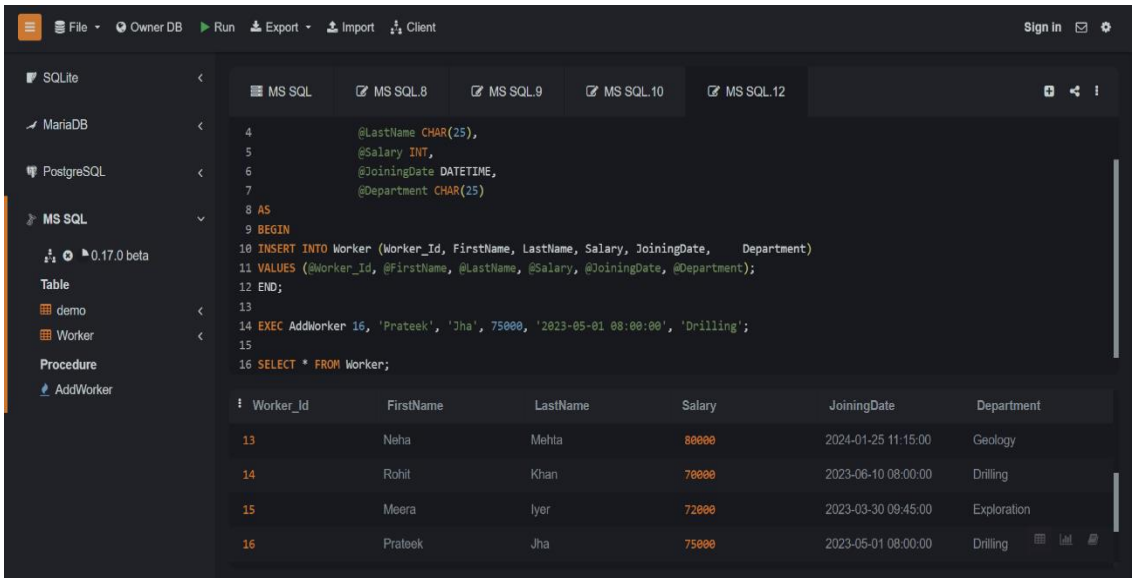
1. **Create a stored procedure that takes in IN parameters for all the columns in the Worker table and adds a new record to the table and then invokes the procedure call.**
    a. Description: This stored procedure inserts a new worker record into the Worker table.
    b. Advantage: Standardizes the process of adding new records and ensures data integrity.
    c. Query:

```
CREATE PROCEDURE AddWorker
    @Worker_Id INT,
    @FirstName CHAR(25),
    @LastName CHAR(25),
    @Salary INT,
    @JoiningDate DATETIME,
    @Department CHAR(25)
AS
BEGIN
    INSERT INTO Worker (Worker_Id, FirstName, LastName, Salary,
JoiningDate, Department)
    VALUES (@Worker_Id, @FirstName, @LastName, @Salary,
@JoiningDate, @Department);
END;
```

->Procedure Call:

```
EXEC AddWorker 16, 'Prateek', 'Jha', 75000, '2023-05-01
08:00:00', 'Drilling';

SELECT * FROM Worker;
```

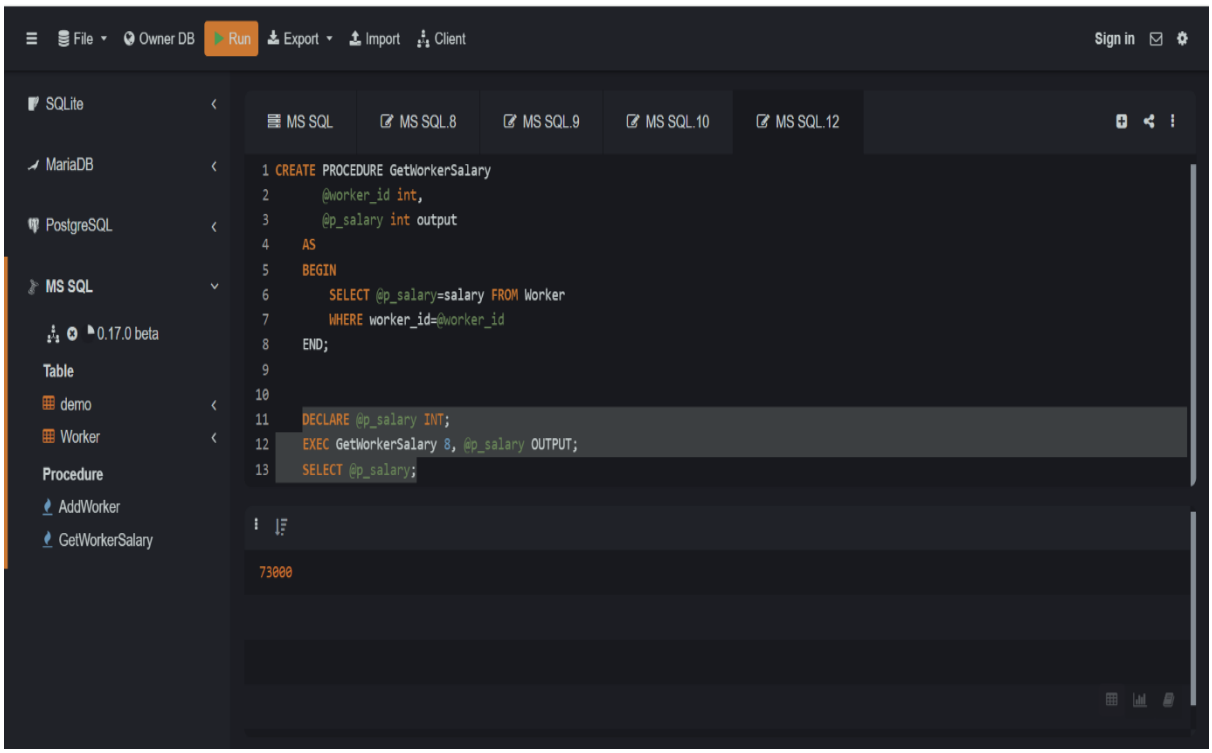


2. **Write stored procedure takes in an IN parameter for WORKER_ID and an OUT parameter for SALARY. It should retrieve the salary of the worker with the given ID and returns it in the p_salary parameter. Then make the procedure call.**
    a. Description: This stored procedure retrieves the salary of a worker based on their worker ID.
    b. Advantage: Provides a secure and efficient way to access sensitive salary information.
    c. Query:

```
CREATE PROCEDURE GetWorkerSalary
    @worker_id int,
    @p_salary int output
AS
BEGIN
    SELECT @p_salary=salary from Worker
    WHERE worker_id=@worker_id
END;
```

->Procedure Call:

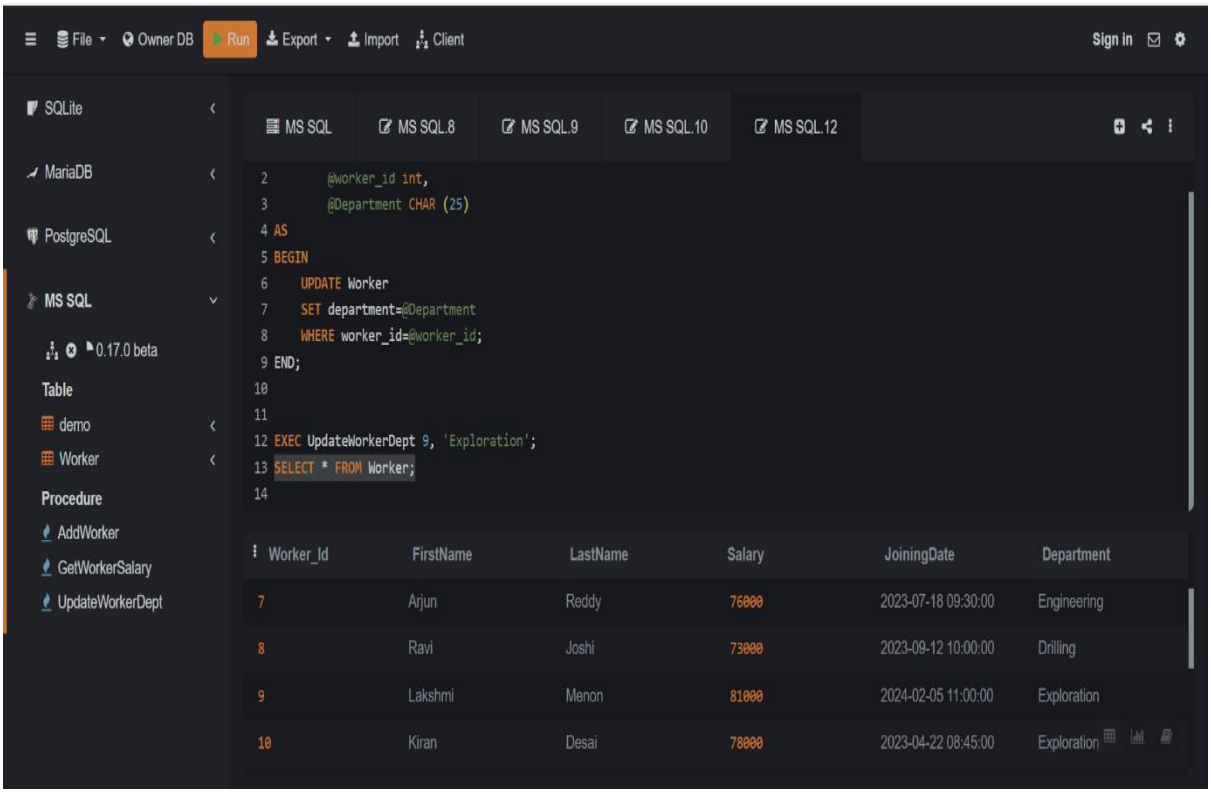
```
DECLARE @p_salary INT;
EXEC GetWorkerSalary 8, @p_salary OUTPUT;
SELECT @p_salary;
```



3. **Create a stored procedure that takes in IN parameters for WORKER_ID and DEPARTMENT. It should update the department of the worker with the given ID. Then make a procedure call.**
   a. Description: This stored procedure updates the department of a worker based on their worker ID.
   b. Advantage: Ensures that department changes are consistently applied across the database.
   c. Query:

```
CREATE PROCEDURE UpdateWorkerDept
    @worker_id int,
    @Department CHAR (25)
AS
BEGIN
    UPDATE Worker
    SET department=@Department
    WHERE worker_id=@worker_id;
END;

->Procedure Call:
    EXEC UpdateWorkerDept 9, 'Exploration';
    SELECT * FROM Worker;
```

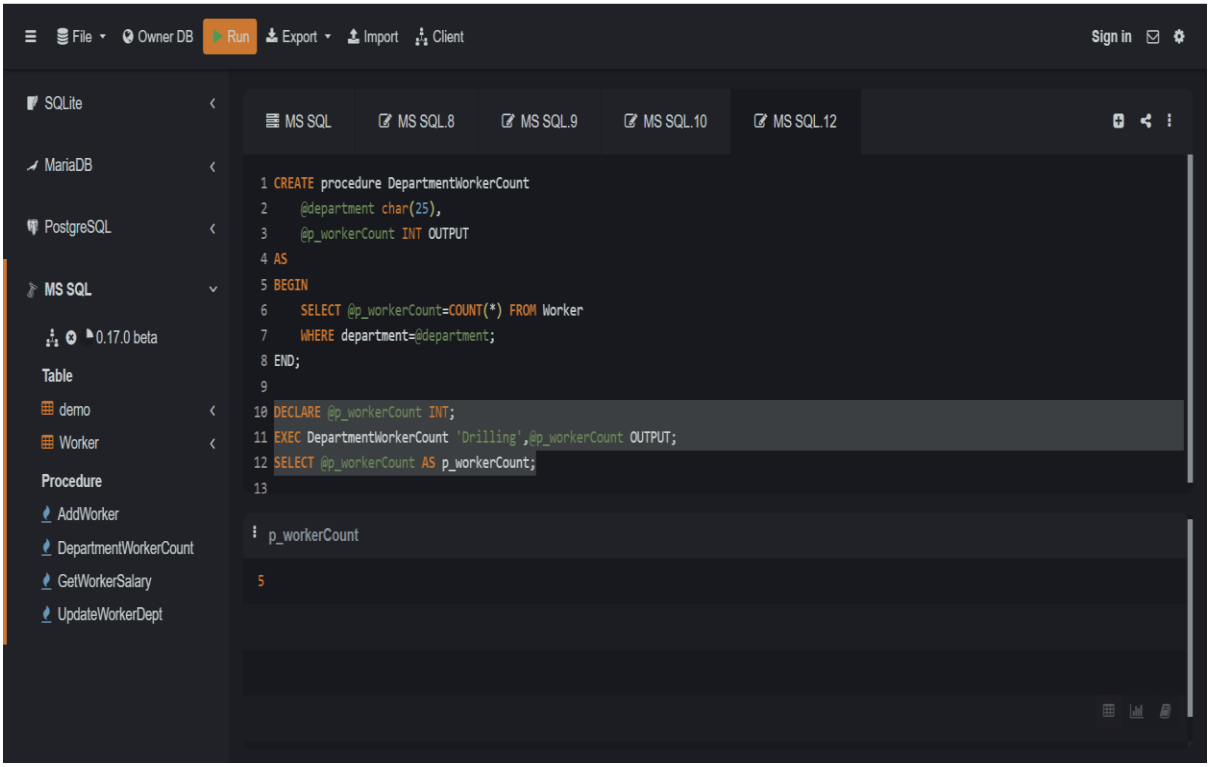


4. **Write a stored procedure that takes in an IN parameter for DEPARTMENT and an OUT parameter for p_workerCount. It should retrieve the number of workers in the given department and returns it in the p_workerCount parameter. Make procedure call.**
    a. Description: This stored procedure retrieves the number of workers in a specified department.
    b. Advantage: Helps in workforce planning and departmental analysis.
    c. Query:

```
CREATE procedure DepartmentWorkerCount
        @department char(25),
        @p_workerCount INT OUTPUT
    AS
    BEGIN
        SELECT @p_workerCount=count(*) from Worker
        WHERE department=@department;
    END;
```

->Procedure Call:

```
DECLARE @p_workerCount INT;
EXEC DepartmentWorkerCount 'Drilling',@p_workerCount OUTPUT;
SELECT @p_workerCount as p_workerCount;
```



5. **Write a stored procedure that takes in an IN parameter for DEPARTMENT and an OUT parameter for p_avgSalary. It should retrieve the average salary of all workers in the given department and returns it in the p_avgSalary parameter and call the procedure.**
    a. Description: This stored procedure calculates the average salary of all workers in a specified department.
    b. Advantage: Facilitates financial planning and budget allocation within departments.
    c. Query:

```
CREATE PROCEDURE DepartmentAvgSalary
        @Department CHAR(25),
        @p_avgSalary DECIMAL(10,2) OUTPUT
AS
BEGIN
        SELECT @p_avgSalary=avg(salary) from Worker
        WHERE department=@Department;
END;
```

->Procedure Call:

```
DECLARE @p_avgSalary DECIMAL(10,2);
EXEC DepartmentAvgSalary 'Engineering',@p_avgSalary OUTPUT;
SELECT @p_avgSalary as p_avgSalary;
```