

STORED PROCEDURES

➤ CREATED TABLE **worker**

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' list, with 'entri_dsm1_d36' selected. The main editor window shows a SQL script for creating the 'worker' table and inserting data. The script is as follows:

```
1 CREATE TABLE worker (
2     PRIMARY KEY,
3     FirstName CHAR(25),
4     LastName CHAR(25),
5     Salary INT(15),
6     JoiningDate DATETIME,
7     Department CHAR(25)
8 );
9
10 INSERT INTO Worker (Worker_Id, FirstName, LastName, Salary, JoiningDate, Department)
11 VALUES
12 (101, 'John', 'Doe', 50000, '2022-01-15', 'HR'),
13 (102, 'Jane', 'Smith', 60000, '2021-03-10', 'Finance'),
14 (103, 'Michael', 'Johnson', 55000, '2020-11-20', 'IT'),
15 (104, 'Emily', 'Davis', 48000, '2019-07-05', 'HR'),
16 (105, 'Robert', 'Brown', 70000, '2023-06-12', 'Marketing');
17
18 SELECT * FROM Worker;
```

The 'Result Grid' at the bottom displays the data inserted into the 'worker' table:

Worker_Id	FirstName	LastName	Salary	JoiningDate	Department
101	John	Doe	50000	2022-01-15 00:00:00	HR
102	Jane	Smith	60000	2021-03-10 00:00:00	Finance
103	Michael	Johnson	55000	2020-11-20 00:00:00	IT
104	Emily	Davis	48000	2019-07-05 00:00:00	HR
105	Robert	Brown	70000	2023-06-12 00:00:00	Marketing

➤ CREATED A STORED PROCEDURE **AddWorker**

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' list, with 'entri_dsm1_d36' selected. The main editor window shows a SQL script for creating the 'AddWorker' stored procedure. The script is as follows:

```
18 SELECT * FROM Worker;
19
20 # 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.
21
22 DELIMITER $$
23
24 CREATE PROCEDURE AddWorker (
25     IN p_Worker_Id INT,
26     IN p_FirstName CHAR(25),
27     IN p_LastName CHAR(25),
28     IN p_Salary INT(15),
29     IN p_JoiningDate DATETIME,
30     IN p_Department CHAR(25)
31 )
32 BEGIN
33     INSERT INTO Worker (Worker_Id, FirstName, LastName, Salary, JoiningDate, Department)
34     VALUES (p_Worker_Id, p_FirstName, p_LastName, p_Salary, p_JoiningDate, p_Department);
35 END $$
36
37 DELIMITER ;
```

The 'Output' window at the bottom displays the execution results:

#	Time	Action	Message	Duration / Fech
1	21:09:34	CREATE TABLE Worker (Worker_Id INT PRIMARY KEY; FirstName CHAR(25); LastName CHAR(25);	0 rows(s) affected. 1 warning(s): 1681 Integer display width is deprecated and will be removed in a future release.	0.032 sec
2	21:09:45	INSERT INTO Worker (Worker_Id, FirstName, LastName, Salary, JoiningDate, Department) VALUES (101, John,	5 row(s) affected. Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
3	21:10:29	SELECT * FROM Worker LIMIT 0. 300	5 row(s) returned	0.000 sec / 0.000 sec
4	21:15:31	SELECT * FROM Worker LIMIT 0. 300	5 row(s) returned	0.000 sec / 0.000 sec
5	21:26:02	CREATE PROCEDURE AddWorker (IN p_Worker_Id INT; IN p_FirstName CHAR(25); IN p_LastName ...	0 rows(s) affected. 1 warning(s): 1681 Integer display width is deprecated and will be removed in a future release.	0.000 sec

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with 'entri_dsm1_d36' selected. The main editor shows a SQL script for creating a stored procedure named 'AddWorker'. The script includes a 'CREATE PROCEDURE' statement with parameters for worker ID, first name, last name, salary, joining date, and department. It then uses an 'INSERT INTO' statement to add a new worker. The script is executed, and the 'Result Grid' shows the output of the procedure call.

```

22 DELIMITER $$
23
24 CREATE PROCEDURE AddWorker (
25     IN p_Worker_Id INT,
26     IN p_FirstName CHAR(25),
27     IN p_LastName CHAR(25),
28     IN p_Salary INT(15),
29     IN p_JoiningDate DATETIME,
30     IN p_Department CHAR(25)
31 )
32 BEGIN
33     INSERT INTO Worker (Worker_Id, FirstName, LastName, Salary, JoiningDate, Department)
34     VALUES (p_Worker_Id, p_FirstName, p_LastName, p_Salary, p_JoiningDate, p_Department);
35 END $$
36
37 DELIMITER ;
38
39 CALL AddWorker(106, 'Alice', 'Williams', 62000, '2024-09-16', 'IT');
40
41

```

Worker_Id	FirstName	LastName	Salary	JoiningDate	Department
101	John	Doe	50000	2022-01-15 00:00:00	HR
102	Jane	Smith	60000	2021-03-10 00:00:00	Finance
103	Michael	Johnson	55000	2020-11-20 00:00:00	IT
104	Emily	Davis	48000	2019-07-05 00:00:00	HR
105	Robert	Brown	70000	2023-06-12 00:00:00	Marketing
106	Alice	Williams	62000	2024-09-16 00:00:00	IT

➤ CREATED A STORED PROCEDURE **GetWorkerSalary**

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with 'entri_dsm1_d36' selected. The main editor shows a SQL script for creating a stored procedure named 'GetWorkerSalary'. The script includes a 'CREATE PROCEDURE' statement with parameters for worker ID and salary. It then uses a 'SELECT' statement to retrieve the salary of the worker. The script is executed, and the 'Result Grid' shows the output of the procedure call.

```

48 OUT p_Salary INT
49 )
50 BEGIN
51     SELECT Salary INTO p_Salary
52     FROM Worker
53     WHERE Worker_Id = p_Worker_Id;
54 END $$
55
56 DELIMITER ;
57
58 SET @salary = 0;
59 CALL GetWorkerSalary(101, @salary);
60 SELECT @salary;
61

```

Worker_Id	Salary
101	50000

The 'Action Output' pane shows the execution details of the stored procedure call:

#	Time	Action	Message	Duration / Fetch
6	21:28:15	CALL AddWorker(106, 'Alice', 'Williams', 62000, '2024-09-16', 'IT')	1 row(s) affected	0.016 sec
7	21:28:21	SELECT * FROM Worker LIMIT 0, 300	6 row(s) returned	0.000 sec / 0.000 sec
8	21:42:38	CREATE PROCEDURE GetWorkerSalary (IN p_Worker_Id INT, OUT p_Salary INT) BEGIN SELECT...	0 row(s) affected	0.016 sec
9	21:44:32	SET @salary = 0	0 row(s) affected	0.000 sec
10	21:44:36	CALL GetWorkerSalary(101, @salary)	1 row(s) affected	0.000 sec
11	21:44:42	SELECT @salary LIMIT 0, 300	1 row(s) returned	0.000 sec / 0.000 sec

➤ CREATED A STORED PROCEDURE

UpdateWorkerDepartment

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'entri_dsm1_d36' selected. The main editor window contains the following SQL code:

```
63 # It should update the department of the worker with the given ID. Then make a procedure call.
64
65 DELIMITER $$
66
67 CREATE PROCEDURE UpdateWorkerDepartment (
68     IN p_Worker_Id INT,
69     IN p_Department CHAR(25)
70 )
71 BEGIN
72     UPDATE Worker
73     SET Department = p_Department
74     WHERE Worker_Id = p_Worker_Id;
75 END $$
76
77 DELIMITER ;
78
79 CALL UpdateWorkerDepartment(101, 'Sales');
80
81 SELECT * FROM Worker;
```

The 'Result Grid' at the bottom shows the output of the 'SELECT * FROM Worker;' query:

Worker_Id	Firstname	Lastname	Salary	JoiningDate	Department
101	John	Doe	50000	2022-01-15 00:00:00	Sales
102	Jane	Smith	60000	2021-03-10 00:00:00	Finance
103	Michael	Johnson	55000	2020-11-20 00:00:00	IT
104	Emily	Davis	48000	2019-07-05 00:00:00	HR
105	Robert	Brown	70000	2023-06-12 00:00:00	Robert
106	Alice	Williams	62000	2024-09-16 00:00:00	IT

➤ CREATED A STORED PROCEDURE

GetWorkerCount

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'entri_dsm1_d36' selected. The main editor window contains the following SQL code:

```
81 SELECT * FROM Worker;
82
83 # 4. Write a stored procedure that takes in an IN parameter for DEPARTMENT and an OUT parameter for p_workerCount.
84 # It should retrieve the number of workers in the given department and returns it in the p_workerCount parameter. Make procedure call.
85
86 DELIMITER $$
87
88 CREATE PROCEDURE GetWorkerCount (
89     IN p_Department CHAR(25),
90     OUT p_workerCount INT
91 )
92 BEGIN
93     SELECT COUNT(*) INTO p_workerCount
94     FROM Worker
95     WHERE Department = p_Department ;
96 END $$
97
98 DELIMITER ;
99
100 SET @workercount = 0;
101 CALL GetWorkerCount ('IT', @workercount);
102 SELECT @workercount;
```

The 'Result Grid' at the bottom shows the output of the 'SELECT @workercount;' query:

@workercount
2

➤ CREATED A STORED PROCEDURE

GetAverageSalary

The screenshot displays the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a tree view of databases, with 'entri_dsml_d36' selected. The main editor window contains SQL code for creating and calling a stored procedure. The code is as follows:

```
104 # 5. Write a stored procedure that takes in an IN parameter for DEPARTMENT and an OUT parameter for p_avgSalary.  
105 #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.  
106  
107 DELIMITER $$  
108  
109 CREATE PROCEDURE GetAverageSalary (  
110     IN p_Deptament CHAR(25),  
111     OUT p_avgSalary DECIMAL(15, 0)  
112 )  
113 BEGIN  
114     SELECT avg(Salary) INTO p_avgSalary  
115     FROM Worker  
116     WHERE Department = p_Deptament ;  
117 END $$  
118  
119 DELIMITER ;  
120  
121 SET @avgSalary = 0;  
122 CALL GetAverageSalary('IT', @avgSalary);  
123 SELECT @avgSalary;  
124  
125  
126
```

Below the code editor, the 'Result Grid' shows the output of the SQL execution:

@avgSalary
58500.00

The bottom status bar indicates 'Result 9' and 'Read Only'.