

## Assignment 7

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of databases: bank, employee, sakila, student, sys, and world. The 'employee' database is selected, showing its tables: departments, employees, and views. The main editor displays a SQL query:

```
3 -- BEGIN
4 -- DECLARE done INT DEFAULT 0;
5 -- DECLARE e_name VARCHAR(100);
6 -- DECLARE cur CURSOR FOR
7 --     SELECT emp_name FROM employees WHERE dept_id = did;
8
9 -- DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
10
11 -- OPEN cur;
12
13 read_loop: LOOP
14     FETCH cur INTO e_name;
15     IF done THEN
16         LEAVE read_loop;
17     END IF;
18     SELECT * FROM employees WHERE emp_name = e_name;
```

Below the query, the 'Result Grid' shows the results of the query. The first result is a table with one row and one column:

Employee
employee_1

The bottom status bar indicates 'Table: departments'.

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of databases: bank, employee, sakila, student, sys, and world. The 'employee' database is selected, showing its tables: departments, employees, and views. The main editor displays a SQL query:

```
1 -- DELIMITER $$
2 -- CREATE PROCEDURE safe_add_employee(IN ename VARCHAR(100), IN sal DECIMAL(10,2), IN did INT)
3 -- BEGIN
4 --     DECLARE dept_exists INT;
5 --     SELECT COUNT(*) INTO dept_exists FROM departments WHERE dept_id = did;
6 --     IF dept_exists = 0 THEN
7 --         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid Department ID';
8 --     ELSE
9 --         INSERT INTO employees (emp_name, salary, dept_id)
10 --         VALUES (ename, sal, did);
11 --         SELECT CONCAT('Employee ', ename, ' added safely.') AS Message;
12 --     END IF;
13 -- END$$
14 -- DELIMITER ;
15
16 CALL safe_add_employee('Grace', 60000, 9);
```

Below the query, the 'Output' pane shows the results of the query. The first result is a table with one row and one column:

Message
Employee Grace added safely.

The bottom status bar indicates 'Table: departments'.

A 'Snipping Tool' window is overlaid on the bottom right, displaying the text: 'Screenshot copied to clipboard. Automatically saved to screenshots folder. Mark-up and share'.

## Assignment 7

The screenshot displays a database management interface with a left-hand 'Navigator' pane and a main script editor area.

**Navigator Pane:** Shows a tree view of database schemas. The 'handler\_lab' schema is expanded, revealing a table named 'employees'.

**Script Editor:** Contains a SQL script for creating and executing a stored procedure. The script is as follows:

```
1  -- DELIMITER $$
2  -- CREATE PROCEDURE delete_employee(IN eid INT)
3  -- BEGIN
4  --     DECLARE rows_affected INT;
5
6  --     DELETE FROM employees WHERE emp_id = eid;
7  --     SET rows_affected = ROW_COUNT();
8
9  --     IF rows_affected = 0 THEN
10 --         SELECT 'No such employee' AS Message;
11 --     ELSE
12 --         SELECT CONCAT('Employee ID ', eid, ' deleted successfully') AS Message;
13 --     END IF;
14 -- END$$
15 -- DELIMITER ;
```

**Execution Results:** Below the script editor, the 'Result Grid' is visible. It shows a single row of data:

Message
Employee ID 2 deleted successfully

## Assignment 7

The screenshot displays a database management interface with a left-hand 'Navigator' pane and a main query editor on the right. The 'Navigator' pane shows a tree structure of databases, including 'bank', 'employee', 'handler\_lab', 'sakila', 'student', 'sys', and 'world'. The 'employee' database is expanded, showing 'Tables', 'Views', 'Stored Procedures', and 'Functions'. The 'handler\_lab' database is also expanded, showing 'Tables' and 'Views'. The 'employees' table is selected under 'handler\_lab'.

The main query editor shows two SQL queries. The first query is a stored procedure named 'count\_employees()' that counts the number of employees in each department and returns the result as a message. The second query is a function named 'annual\_salary()' that calculates the annual salary for a given monthly salary.

**Query 1: count\_employees()**

```
1  -- DELIMITER $$
2  -- CREATE PROCEDURE count_employees()
3  -- BEGIN
4  --     DECLARE done INT DEFAULT 0;
5  --     DECLARE emp_count INT DEFAULT 0;
6  --     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
7
8  --     SELECT COUNT(*) INTO emp_count FROM employees WHERE dept_id IS NOT NULL;
9  --     SELECT CONCAT('Employees with department:', emp_count) AS Message;
10 -- END$$
11 -- DELIMITER ;
12
13 • CALL count_employees();
```

**Result Grid:**

Message
Employees with department:11

**Query 2: annual\_salary(5000)**

```
1  -- DELIMITER $$
2  -- CREATE FUNCTION annual_salary(monthly DECIMAL(10,2))
3  -- RETURNS DECIMAL(10,2)
4  -- DETERMINISTIC
5  -- BEGIN
6  --     RETURN monthly * 12;
7  -- END$$
8  -- DELIMITER ;
9
10 • SELECT annual_salary(5000);
11
12
```

**Result Grid:**

annual_salary(5000)
60000.00

The interface includes a toolbar at the top with icons for file operations, a 'Limit to 1000 rows' dropdown, and a 'Result Grid' button. The bottom of the interface shows the 'Administration' and 'Schemas' tabs, and the 'Information' pane.

## Assignment 7

The image displays two screenshots of the SQL Server Enterprise Manager interface, showing the execution of SQL scripts and the resulting data grids.

**Top Screenshot:**

- Navigator:** The left pane shows the 'handler\_lab' schema expanded, with 'employees' and 'departments' tables visible.
- SQL Script:** The main pane shows a script to create a stored procedure named 'check\_salary'.

```
1  -- DELIMITER $$
2  -- CREATE PROCEDURE check_salary(IN sal DECIMAL(10,2))
3  -- BEGIN
4  --     DECLARE EXIT HANDLER FOR SQLEXCEPTION
5  --     BEGIN
6  --         SELECT 'Salary check failed: Below minimum allowed' AS Message;
7  --     END;
8
9  --     IF sal < 1000 THEN
10 --         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Salary too low';
11 --     ELSE
12 --         SELECT CONCAT('Salary is valid: ', sal) AS Message;
13 --     END IF;
14 -- END$$
15 -- DELIMITER ;
```

- Result Grid:** The bottom pane shows a single row with the message 'Salary check failed: Below minimum allowed'.

**Bottom Screenshot:**

- Navigator:** The left pane shows the 'handler\_lab' schema expanded, with 'employees' and 'departments' tables visible.
- SQL Script:** The main pane shows a script to create a stored procedure named 'bulk\_insert'.

```
5  -- DECLARE CONTINUE HANDLER FOR 1062
6  -- BEGIN
7  --     SELECT CONCAT('Duplicate found for employee_', i, ', ', skipped.) AS Info;
8  -- END;
9
10 -- WHILE i <= 5 DO
11 --     INSERT INTO employees (emp_name, salary, dept_id)
12 --     VALUES (CONCAT('employee_', i), 40000 + (i * 1000), (i % 5) + 1);
13 --     SET i = i + 1;
14 -- END WHILE;
15
16 -- SELECT 'Bulk insert completed' AS Status;
17 -- END$$
18 -- DELIMITER ;
```

- Result Grid:** The bottom pane shows a single row with the status 'Bulk insert completed'.

## Assignment 7

The image shows two screenshots of a database management tool interface, likely MySQL Workbench, demonstrating SQL execution.

**Top Screenshot:**

- Navigator:** Shows the database structure. The **handler\_lab** database is selected, and the **employees** table is highlighted.
- SQL Editor:** Contains the following SQL code:

```
3 -- BEGIN
4 -- DECLARE EXIT HANDLER FOR 1062
5 -- BEGIN
6 --     SELECT 'Employee already exists' AS Message;
7 --     END;
8
9 -- INSERT INTO employees (emp_name, salary, dept_id)
10 -- VALUES (ename, sal, did);
11
12 -- SELECT CONCAT('Employee ', ename, ' added successfully') AS Message;
13 -- END$$
14 -- DELIMITER ;
15
16 • CALL add_employee('John', 55000, 2);
17
```
- Result Grid:** Shows the output of the SQL execution:

Message
Employee John added successfully

**Bottom Screenshot:**

- Navigator:** Shows the database structure. The **handler\_lab** database is selected, and the **departments** table is highlighted.
- SQL Editor:** Contains the following SQL code:

```
1 -- DELIMITER $$
2 -- CREATE PROCEDURE add_departmentt(IN dname VARCHAR(50))
3 -- BEGIN
4 --     DECLARE EXIT HANDLER FOR 1062
5 --     BEGIN
6 --         SELECT 'Department already exists' AS Message;
7 --     END;
8
9 --     INSERT INTO departments (dept_name) VALUES (dname);
10 --     SELECT CONCAT('Department ', dname, ' added successfully') AS Message;
11 -- END$$
12 -- DELIMITER ;
13
14 • call add_departmentt(12);
```
- Result Grid:** Shows the output of the SQL execution:

Message
Department 12 added successfully

The bottom screenshot also shows a **Result 1** tab at the bottom left and a **Read Only** status at the bottom right.