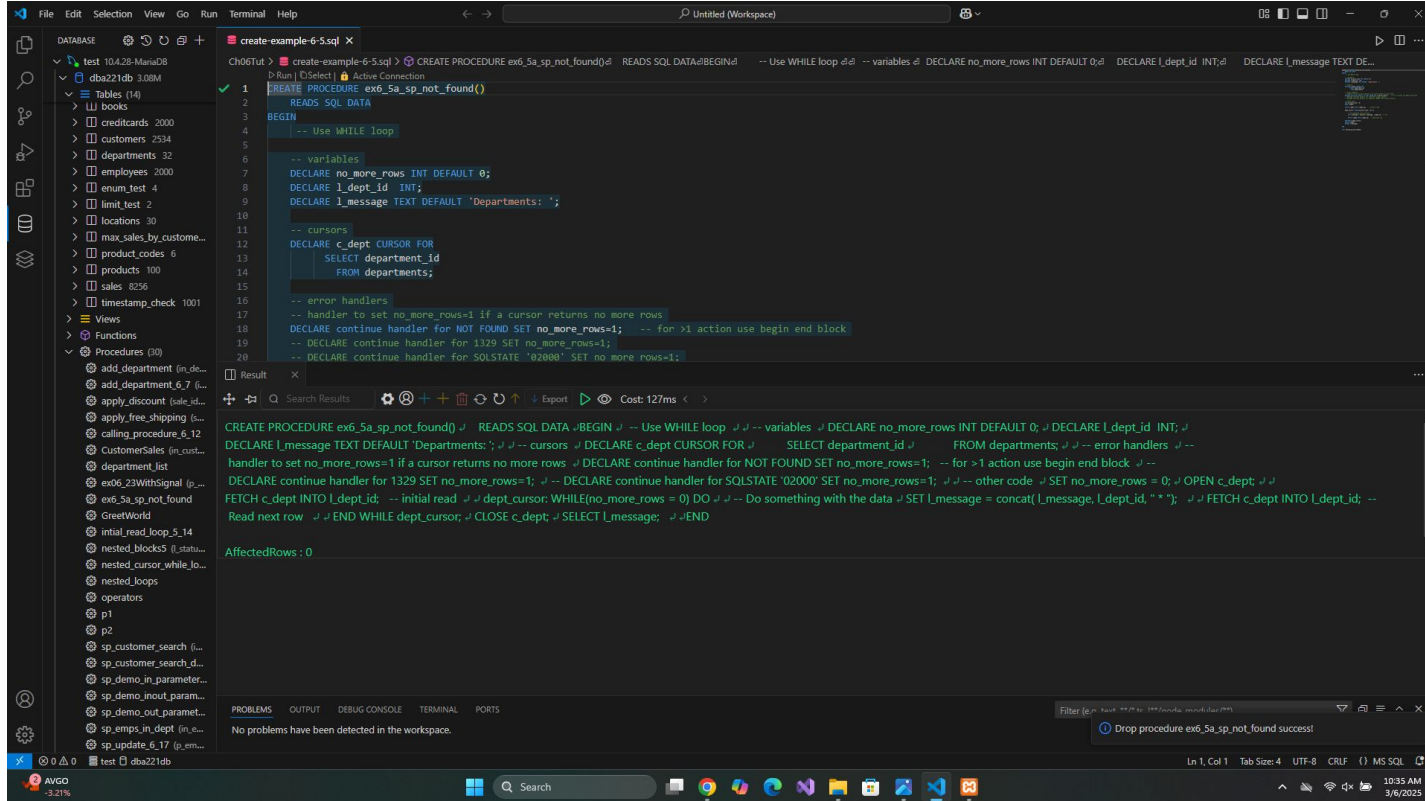


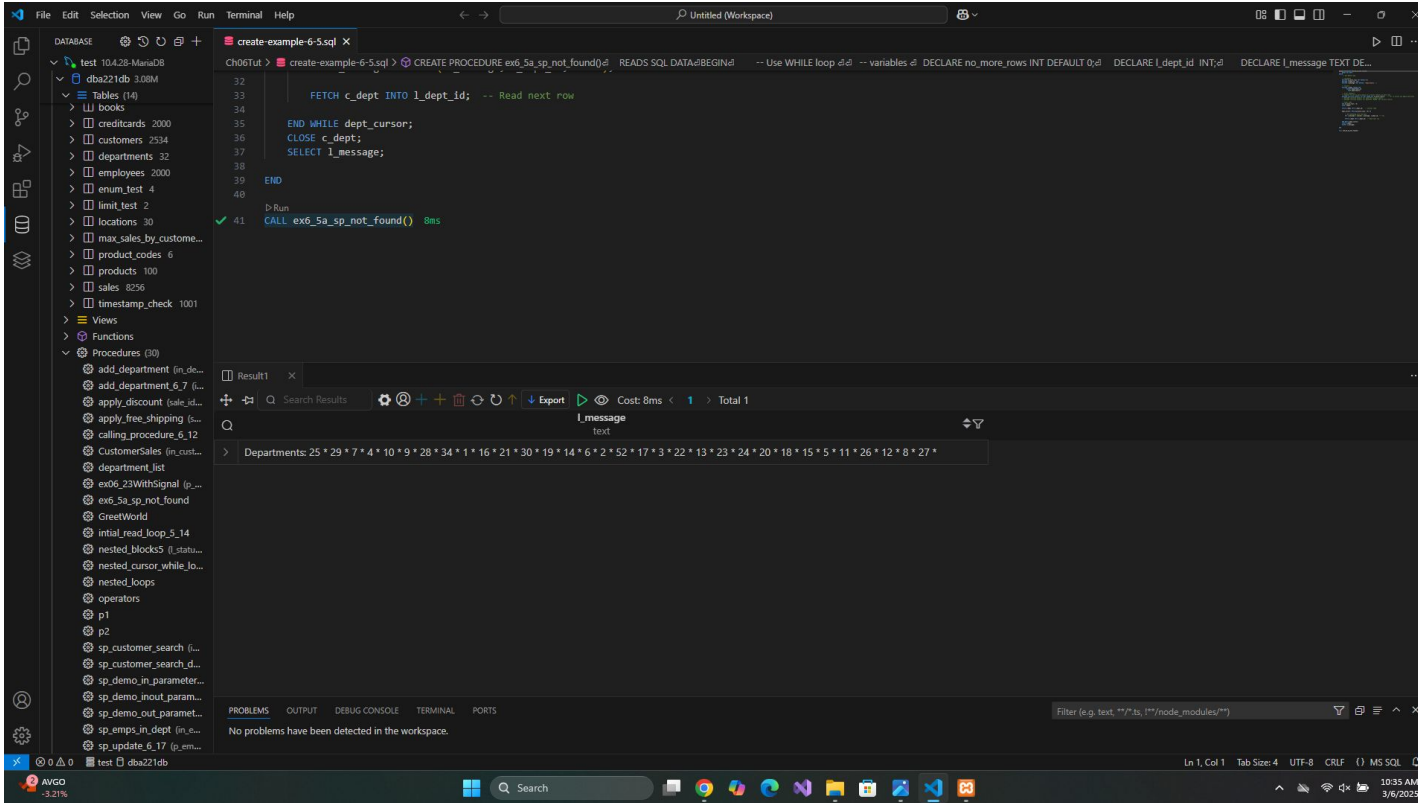
Ch 06 Tutorial

Created by Jack
3/6/2024

Create example 6-5



Call example 6-5



Create example 6-6

The screenshot displays a database IDE interface with a dark theme. On the left, a tree view shows the database structure, including tables, views, and procedures. The main editor window is titled 'create-procedure-template.sql' and contains the following SQL code:

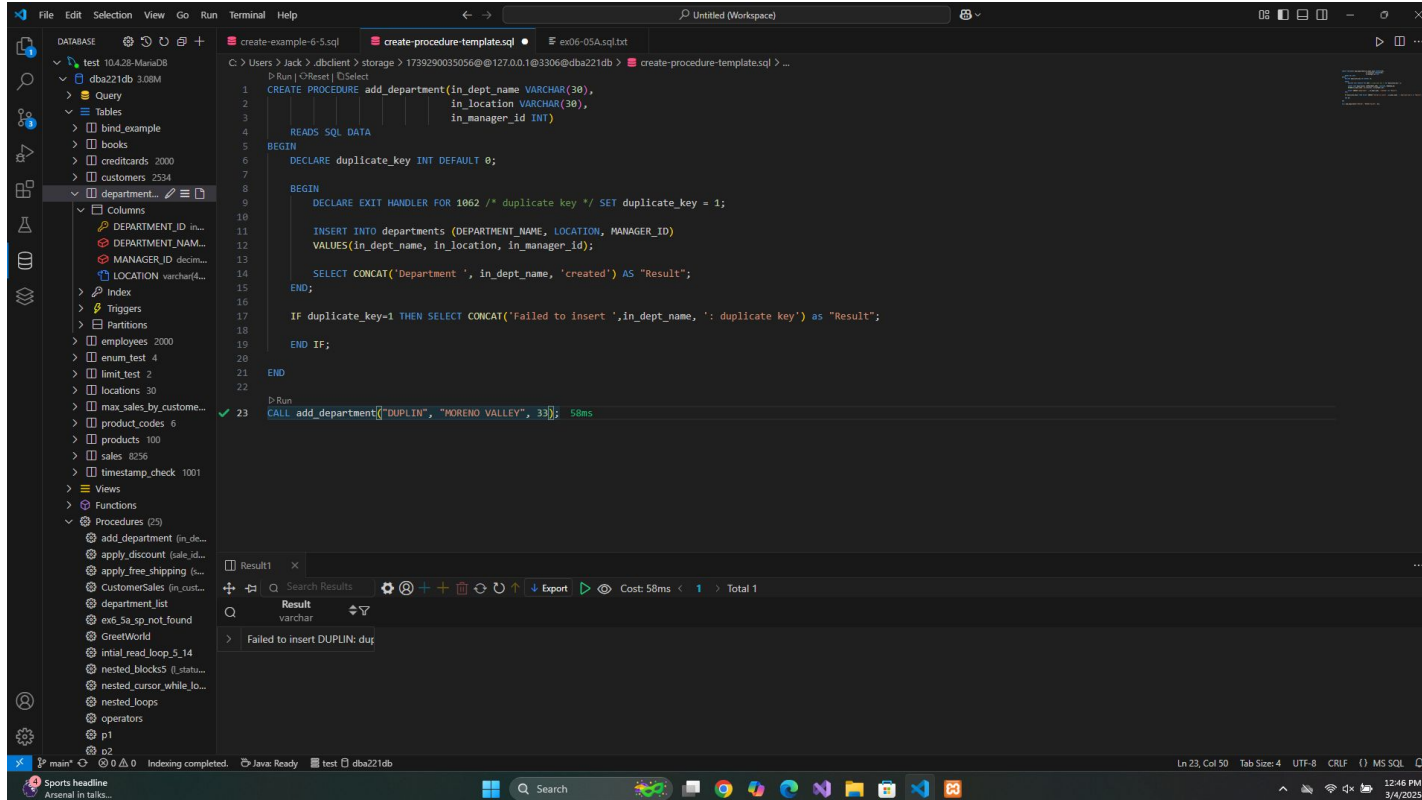
```
1 CREATE PROCEDURE add_department(in_dept_name VARCHAR(30),
2                               in_location VARCHAR(30),
3                               in_manager_id INT)
4   READS SQL DATA
5 BEGIN
6   DECLARE duplicate_key INT DEFAULT 0;
7
8   BEGIN
9     DECLARE EXIT HANDLER FOR 1062 /* duplicate key */ SET duplicate_key = 1;
10
11    INSERT INTO departments (DEPARTMENT_NAME, LOCATION, MANAGER_ID)
12    VALUES(in_dept_name, in_location, in_manager_id);
13
14    SELECT CONCAT('Department ', in_dept_name, ' created') AS "Result";
15  END;
16
17  IF duplicate_key=1 THEN SELECT CONCAT("Failed to insert ',in_dept_name, ': duplicate key") as "Result";
18
19  END IF;
20
21 END
```

The code is executed, and the results pane at the bottom shows the following output:

```
CREATE PROCEDURE add_department(in_dept_name VARCHAR(30), in_location VARCHAR(30), in_manager_id INT)  READS SQL DATA
DECLARE EXIT HANDLER FOR 1062 /* duplicate key */ SET duplicate_key = 1;
INSERT INTO departments (DEPARTMENT_NAME, LOCATION, MANAGER_ID) VALUES(in_dept_name, in_location, in_manager_id);
SELECT CONCAT('Department ', in_dept_name, ' created') AS "Result";
END;
IF duplicate_key=1 THEN SELECT CONCAT("Failed to insert ',in_dept_name, ': duplicate key") as "Result";
END IF;
END
```

The results pane also indicates 'AffectedRows: 0' and 'Cost: 142ms'.

Call example 6-6



The screenshot shows an IDE with a dark theme. On the left is a database explorer showing a schema with tables like 'bind_example', 'books', 'creditcards', 'customers', 'department...', 'employees', 'enum_test', 'limit_test', 'locations', 'max_sales_by_custom...', 'product_codes', 'products', 'sales', 'timestamp_check', 'views', 'functions', and 'procedures'. The 'department...' table is selected. The main editor displays a SQL script in a file named 'create-procedure-template.sql'. The script defines a procedure 'add_department' that takes department name, location, and manager ID as inputs. It includes a duplicate key check and an insert statement. Below the script, the execution results are shown. A green checkmark indicates the procedure was created successfully. The next line shows the procedure being called with 'DUPLIN' as the department name, 'MORENO VALLEY' as the location, and '33' as the manager ID. The execution time is 58ms. Below the results, a message states 'Failed to insert DUPLIN: dup'. The status bar at the bottom shows 'Ln 23, Col 50', 'Tab Size: 4', 'UTF-8', 'CRLF', and 'MS SQL'.

```
1 CREATE PROCEDURE add_department(in_dept_name VARCHAR(30),
2                                in_location VARCHAR(30),
3                                in_manager_id INT)
4
5 READS SQL DATA
6
7 BEGIN
8     DECLARE duplicate_key INT DEFAULT 0;
9
10    BEGIN
11        DECLARE EXIT HANDLER FOR 1062 /* duplicate key */ SET duplicate_key = 1;
12
13        INSERT INTO departments (DEPARTMENT_NAME, LOCATION, MANAGER_ID)
14        VALUES(in_dept_name, in_location, in_manager_id);
15
16        SELECT CONCAT('Department ', in_dept_name, 'created') AS "Result";
17    END;
18
19    IF duplicate_key=1 THEN SELECT CONCAT('Failed to insert ',in_dept_name, ': duplicate key') as "Result";
20    END IF;
21
22 END
23
24 CALL add_department('DUPLIN', 'MORENO VALLEY', 33); 58ms
```

Result1

Result
varchar
Failed to insert DUPLIN: dup

Call example 6-6

The screenshot displays an IDE window with a SQL editor and a results pane. The SQL editor contains a procedure named `add_department` that inserts a new department into the `departments` table. The procedure includes a `DECLARE` statement for `duplicate_key`, an `EXIT HANDLER` for error 1062, and an `INSERT` statement. The results pane shows the output of the procedure call, indicating that the department `CUSTOMER SATISFACTION` was successfully created.

```
1 CREATE PROCEDURE add_department(in_dept_name VARCHAR(30),
2                               in_location VARCHAR(30),
3                               in_manager_id INT)
4 READS SQL DATA
5 BEGIN
6     DECLARE duplicate_key INT DEFAULT 0;
7
8     BEGIN
9         DECLARE EXIT HANDLER FOR 1062 /* duplicate key */ SET duplicate_key = 1;
10
11         INSERT INTO departments (DEPARTMENT_NAME, LOCATION, MANAGER_ID)
12         VALUES(in_dept_name, in_location, in_manager_id);
13
14         SELECT CONCAT('Department ', in_dept_name, 'created') AS "Result";
15     END;
16
17     IF duplicate_key=1 THEN SELECT CONCAT('Failed to insert ',in_dept_name, ': duplicate key') as "Result";
18     END IF;
19 END IF;
20
21 END
22
23 P>Run
24 CALL add_department('DUPLIN', 'MORENO VALLEY', 33);
25
26 P>Run
27 CALL add_department('CUSTOMER SATISFACTION','DAVIS',4); 74ms
```

Result1

Result
Department CUSTOMER SATISFACTIONcreated

Create example 6-7

The screenshot displays a SQL IDE interface with a dark theme. On the left, a database explorer shows a schema named 'test' containing various tables and views. The 'Procedures' folder is expanded, showing a list of procedures including 'add_department', 'add_department_6_7', and others. The main editor window is titled 'create-procedure-template.sql' and contains the following SQL code:

```
1 CREATE PROCEDURE add_department_6_7(in_dept_name VARCHAR(30),
2                                     in_location VARCHAR(30),
3                                     in_manager_id INT)
4
5     MODIFIES SQL DATA
6
7 BEGIN
8     DECLARE duplicate_key INT DEFAULT 0;
9
10    DECLARE CONTINUE HANDLER FOR 1062 /* duplicate key */
11        SET duplicate_key = 1;
12
13    INSERT INTO departments(DEPARTMENT_NAME, LOCATION, MANAGER_ID)
14    VALUES (in_dept_name, in_location, in_manager_id);
15
16    IF duplicate_key = 1 THEN
17        SELECT CONCAT('Failed to insert ', in_dept_name, ': duplicate key') AS "Result";
18    ELSE
19        SELECT CONCAT('Department ', in_dept_name, ' created ') AS "Result";
20    END IF;
21
22 END
```

Below the code editor, the 'Result' pane shows the execution output:

```
CREATE PROCEDURE add_department_6_7(in_dept_name VARCHAR(30), in_location VARCHAR(30), in_manager_id INT) MODIFIES SQL DATA BEGIN DECLARE duplicate_key INT DEFAULT 0;
DECLARE CONTINUE HANDLER FOR 1062 /* duplicate key */ SET duplicate_key = 1; INSERT INTO departments(DEPARTMENT_NAME, LOCATION, MANAGER_ID) VALUES (in_dept_name, in_location, in_manager_id);
IF duplicate_key = 1 THEN SELECT CONCAT('Failed to insert ', in_dept_name, ': duplicate key') AS "Result"; ELSE SELECT CONCAT('Department ', in_dept_name, ' created ') AS "Result"; END IF; END
```

The 'AffectedRows' section shows 0 rows affected. At the bottom right, a status message indicates: 'Drop procedure add_department_6_7 success!'. The bottom status bar shows 'Ln 13, Col 1', 'Tab Size: 4', 'UTF-8', 'CR/LF', and 'MS SQL'.

Call example 6-7

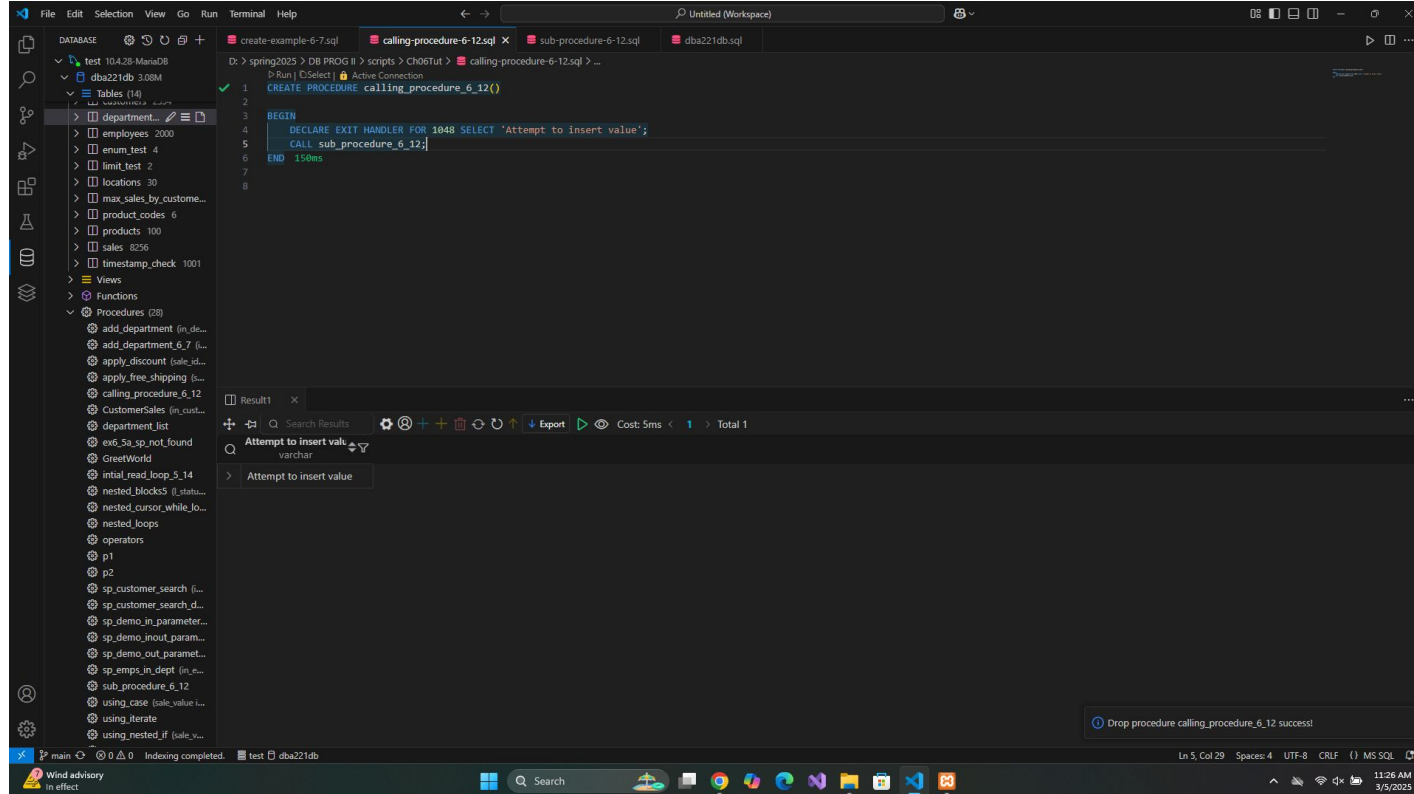
The screenshot displays the SQL Developer interface with a workspace titled 'Untitled (Workspace)'. The left sidebar shows a database connection to 'dba221db' with a tree view of tables and procedures. The main editor window contains a PL/SQL script for a procedure named 'add_department_6_7'. The script includes a 'DECLARE CONTINUE HANDLER FOR 1062' block to handle duplicate key errors, an 'INSERT INTO departments' statement, and an 'IF duplicate_key = 1 THEN' block that returns a message. The procedure is called with the parameters 'DUPLIN', 'MORENO VALLEY', and '33'. The execution results pane at the bottom shows the output of the procedure call, which is 'Failed to insert DUPLIN: duplicate key'.

```
8 DECLARE CONTINUE HANDLER FOR 1062 /* duplicate key */
9 SET duplicate_key = 1;
10
11 INSERT INTO departments(DEPARTMENT_NAME, LOCATION, MANAGER_ID)
12 VALUES (in_dept_name, in_location, in_manager_id);
13
14 IF duplicate_key = 1 THEN
15 SELECT CONCAT('Failed to insert ', in_dept_name, ': duplicate key') AS "Result";
16 ELSE
17 SELECT CONCAT('Department ', in_dept_name, ' created ') AS "Result";
18 END IF;
19 END IF;
20
21 END
22
23 CALL add_department_6_7('DUPLIN', 'MORENO VALLEY', 33); 57ms
```

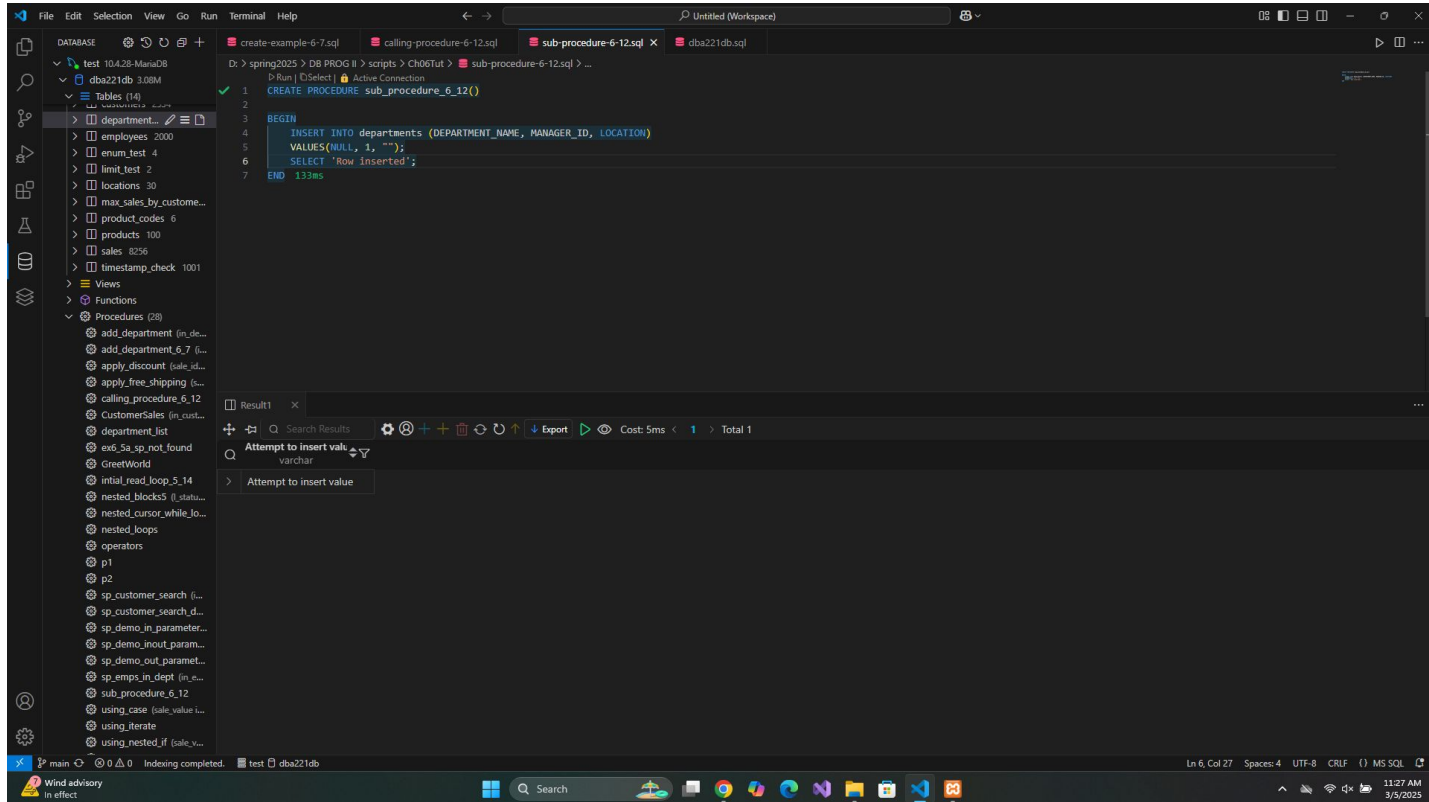
Result1

Result
Failed to insert DUPLIN: duplicate key

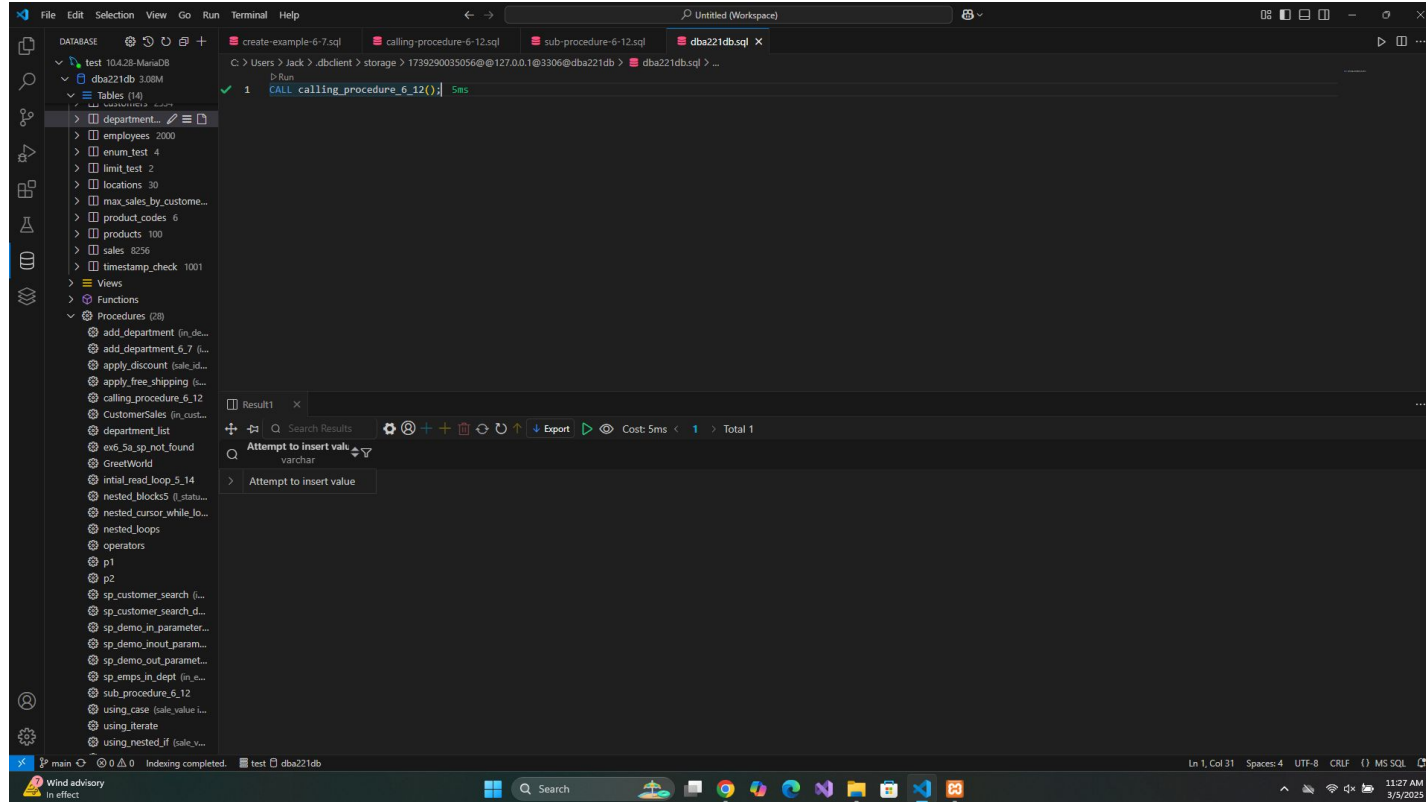
Create calling_procedure 6-12



Create sub procedure 6-12



Call example 6-12



Create example 6-17

The screenshot displays a SQL IDE interface with a dark theme. The left sidebar shows a database explorer for 'test 10.4.28-MariaDB' with a schema 'dba221db'. The main editor window is titled 'create-procedure-template.sql' and contains the following SQL code:

```
1 CREATE PROCEDURE sp_update_6_17(p_employee_id INT,  
2 p_dob DATE)  
3  
4 BEGIN  
5 DECLARE employee_is_too_young CONDITION FOR SQLSTATE '45000';  
6  
7 IF DATE_SUB(curdate(), INTERVAL 16 YEAR) < p_dob THEN  
8 SIGNAL employee_is_too_young  
9 SET MESSAGE_TEXT = 'Employee must be 16 years or older';  
10 ELSE  
11 UPDATE employees  
12 SET DATE_OF_BIRTH = p_dob  
13 WHERE EMPLOYEE_ID = p_employee_id;  
14 END IF;  
15 END
```

Below the editor, the 'Result' tab shows the execution output:

```
CREATE PROCEDURE sp_update_6_17(p_employee_id INT, p_dob DATE) v-BEGIN v-DECLARE employee_is_too_young CONDITION FOR SQLSTATE '45000'; v-IF DATE_SUB(curdate(), INTERVAL 16 YEAR) < p_dob THEN v-SIGNAL employee_is_too_young v-SET MESSAGE_TEXT = 'Employee must be 16 years or older'; v-ELSE v-UPDATE employees v-SET DATE_OF_BIRTH = p_dob v-WHERE EMPLOYEE_ID = p_employee_id; v-END IF; v-END  
AffectedRows : 0
```

The bottom status bar indicates 'Ln 15, Col 4', 'Tab Size: 4', 'UTF-8', 'CRLF', and 'MS SQL'.

Call example 6-17

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'DATABASE' pane shows the 'test' database with various tables, views, and procedures. The 'Procedures (30)' folder is expanded, showing a list of procedures including 'add_department', 'add_department_6_7', 'apply_discount', 'apply_free_shipping', 'calling_procedure_6_12', 'CustomerSales', 'department_list', 'ex06_23WithSignal', 'ex6_5a_sp_not_found', 'GreetWorld', 'initial_read_loop_5_14', 'nested_blocks5', 'nested_cursor_while_lo...', 'nested_loops', 'operators', 'p1', 'p2', 'sp_customer_search', 'sp_customer_search_d...', 'sp_demo_in_parameter...', 'sp_demo_inout_param...', 'sp_demo_out_paramet...', 'sp_emps_in_dept', 'sp_update_6_17', 'sub_procedure_6_12', 'using_case', 'using_iterate', 'using_nested_if', 'using_repeat', and 'using_while_less'.

The main pane shows the 'create-procedure-template.sql' file with the following SQL code:

```
CREATE PROCEDURE sp_update_6_17(p_employee_id INT, p_dob DATE)
AS
BEGIN
    DECLARE employee_is_too_young CONDITION FOR SQLSTATE '45000';
    IF DATE_SUB(curdate(), INTERVAL 16 YEAR) < p_dob THEN
        SIGNAL employee_is_too_young
        SET MESSAGE_TEXT = 'Employee must be 16 years or older';
    ELSE
        UPDATE employees
        SET DATE_OF_BIRTH = p_dob
        WHERE EMPLOYEE_ID = p_employee_id;
    END IF;
END
```

The 'Result' pane shows the execution of the procedure:

```
CALL sp_update_6_17(1, now()); 3ms Employee must be 16 years or older
```

The 'Result' pane also displays the SQL code for the procedure and the 'AffectedRows: 0'.

The bottom status bar shows 'Ln 17, Col 31 Tab Size: 4 UTF-8 CRLF () MS SQL'.

Create example 6-23

The screenshot displays the SQL Server Enterprise Edition interface. On the left, the 'DATABASE' pane shows the 'test' database with various tables and procedures. The 'Procedures' folder is expanded, showing a list of procedures including 'sp_add_department', 'sp_add_department_6_7', 'sp_apply_discount', 'sp_apply_free_shipping', 'sp_calling_procedure_6_12', 'sp_customerSales', 'sp_departmentList', 'sp_ex06_23WithSignal', 'sp_ex6_5a_sp_not_found', 'sp_greetWorld', 'sp_initial_read_loop_5_14', 'sp_nested_blockS', 'sp_nested_cursor_while_loops', 'sp_nested_loops', 'sp_operators', 'sp_p1', 'sp_p2', 'sp_sp_customer_search', 'sp_sp_customer_search_d...', 'sp_sp_demo_in_parameter...', 'sp_sp_demo_inout_param...', 'sp_sp_demo_out_param...', 'sp_sp_emp_in_dept_in_s...', 'sp_sp_update_6_17', 'sp_sub_procedure_6_12', 'sp_using_case', 'sp_using_iterate', 'sp_using_nested_if', 'sp_using_repeat', 'sp_using_while_less', 'sp_dba221sales', 'sp_phpmyadmin', and 'sp_premiere_products'. The 'sp_ex06_23WithSignal' procedure is selected.

The main pane shows the SQL code for the 'sp_ex06_23WithSignal' procedure. The code is as follows:

```
CREATE PROCEDURE sp_ex06_23WithSignal
    @p_department_name VARCHAR(30),
    @p_manager_surname VARCHAR(30),
    @p_manager_firstname VARCHAR(30),
    @p_location VARCHAR(30)
AS
BEGIN
    /* Chapter 6: sp_add_department version Example 6-23 with signal */

    /* START Declare Conditions */
    DECLARE duplicate_key CONDITION FOR 1062;
    DECLARE foreign_key_violated CONDITION FOR 1452; -- changed from 1216 per MySQL reference

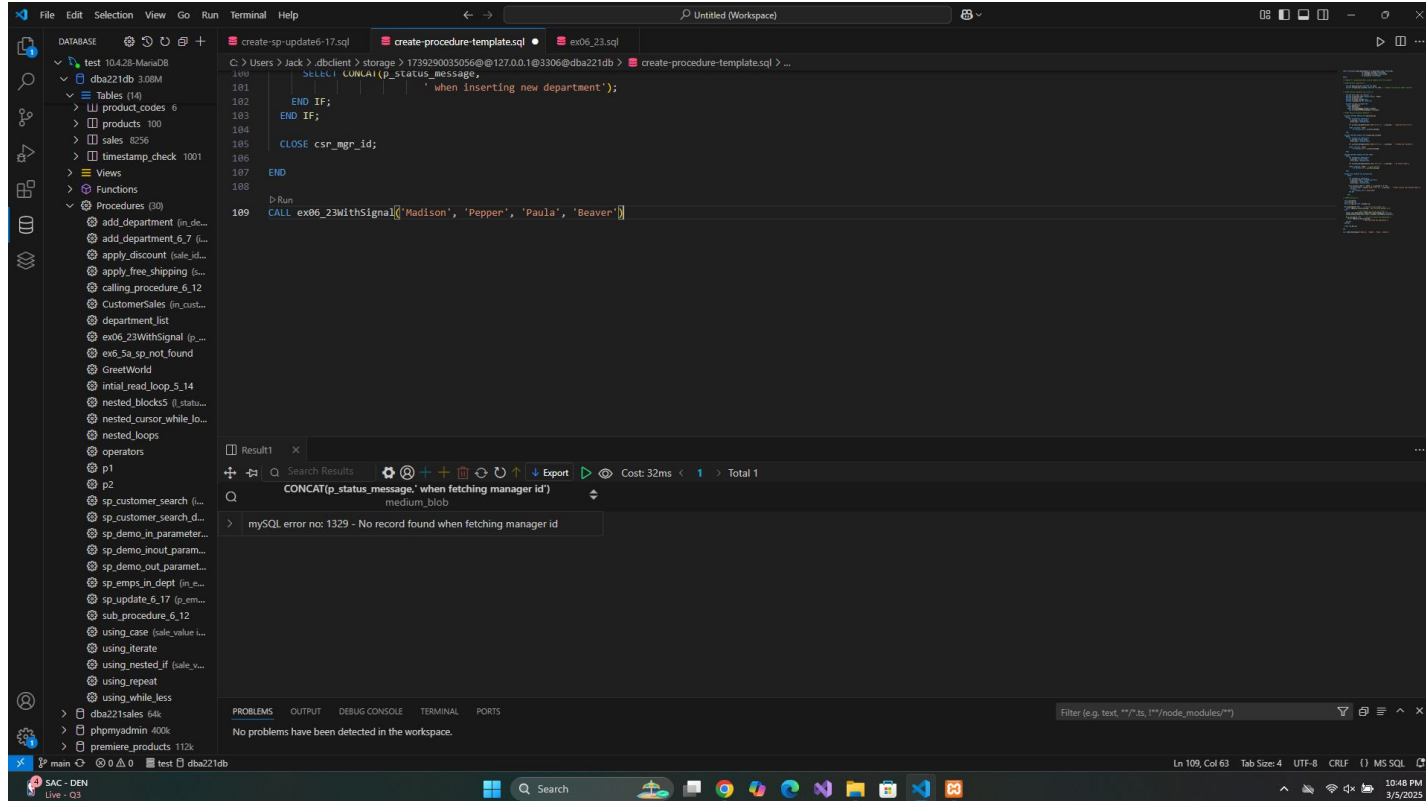
    /* START Declare variables and cursors */
    DECLARE error_text TEXT DEFAULT '';
    DECLARE p_sqlstate_code CHAR(5) DEFAULT '00000';
    DECLARE p_sqlcode INT;
    DECLARE p_status_message TEXT;
    DECLARE l_manager_id INT DEFAULT 0;

    DECLARE csr_mgr_id CURSOR FOR
    SELECT employee_id
    FROM employees
    WHERE surname=UPPER(p_manager_surname)
    AND firstname=UPPER(p_manager_firstname);

    /* START Declare Exception Handlers */
    DECLARE CONTINUE HANDLER FOR duplicate_key
    BEGIN
        GET DIAGNOSTICS CONDITION 1
        p_sqlcode = MYSQL_ERRNO,
        error_text = MESSAGE_TEXT;
        SET p_status_message=concat('MySQL error no:', p_sqlcode, '- Duplicate key error');
        SIGNAL SQLSTATE '01000'
        SET MESSAGE_TEXT = p_status_message;
    END;
    DECLARE CONTINUE HANDLER FOR foreign_key_violated
    BEGIN
        GET DIAGNOSTICS CONDITION 1
        p_sqlcode = MYSQL_ERRNO,
        error_text = MESSAGE_TEXT;
        SET p_status_message=concat('MySQL error no:', p_sqlcode, '- Foreign key violated');
        SIGNAL SQLSTATE '01000'
        SET MESSAGE_TEXT = p_status_message;
    END;
END
```

The bottom pane shows the 'PROBLEMS' tab, which is currently empty, indicating that no problems have been detected in the workspace. A message at the bottom right states: 'Drop procedure ex06_23WithSignal success!'.

Call example 6-23



Call example 6-23

The screenshot displays a MySQL IDE interface with a dark theme. On the left, a 'DATABASE' sidebar shows a tree view of the database structure, including tables, views, and procedures. The main editor window is titled 'create-procedure-template.sql' and contains SQL code for a procedure named 'ex06_23WithSignal'. The code includes a 'SELECT' statement with a 'CONCAT' function and a 'CALL' statement. The 'CALL' statement is highlighted in green, and a green checkmark is visible next to it. Below the code, the 'Result1' tab shows the output of the 'CALL' statement, which is '53ms'. At the bottom, the 'PROBLEMS' tab is active, displaying a message: 'mySQL error no: 1062 - Duplicate key error when inserting new department'. The status bar at the bottom indicates the current file is 'Ln 111, Col 62' and the encoding is 'UTF-8'.

```
CREATE PROCEDURE ex06_23WithSignal(
    IN p_department VARCHAR(50),
    IN p_manager VARCHAR(50),
    IN p_employee VARCHAR(50),
    IN p_department_id INT(11)
)
BEGIN
    SELECT CONCAT(p_status_message, 'when inserting new department');
END IF;
END IF;
CLOSE csr_mgr_id;
END
```

Result1

Cost: 53ms < 1 > Total 1

CONCAT(p_status_message, 'when inserting new department')

medium_blob

mySQL error no: 1062 - Duplicate key error when inserting new department

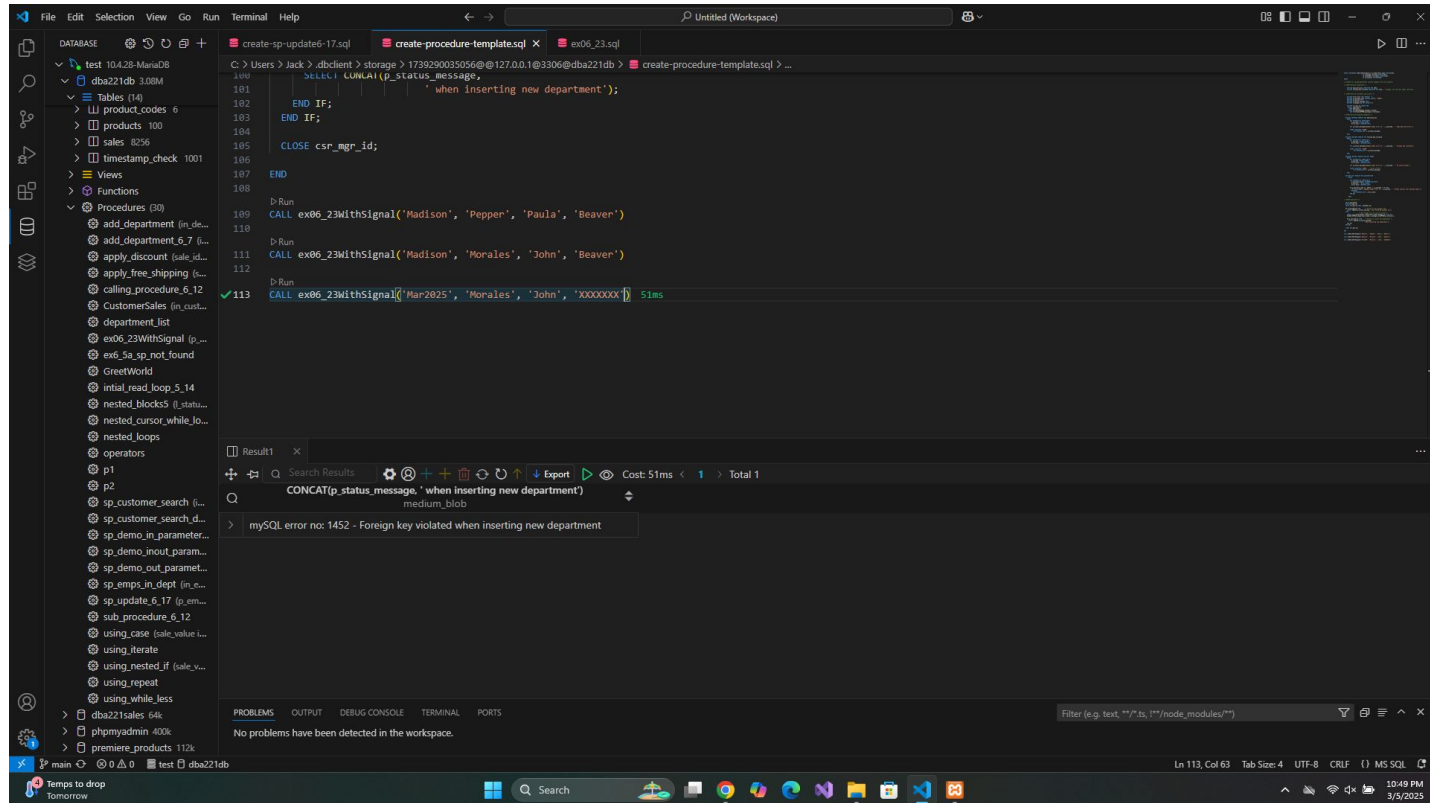
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

No problems have been detected in the workspace.

Filter (e.g. text, ""ts, ""node_modules/"")

Ln 111, Col 62 Tab Size: 4 UTF-8 CRLF () MS SQL

Call example 6-23



Call example 6-23

The screenshot shows a Visual Studio Code editor with a MySQL database project. The left sidebar displays the database structure, including tables, views, and procedures. The main editor area shows a SQL file named `create-procedure-template.sql` with the following content:

```
110 SELECT CONCAT(p_status_message,  
101 'when inserting new department');  
102  
103 END IF;  
104  
105 END IF;  
106  
107 CLOSE csr_mgr_id;  
108  
109  
110  
111  
112  
113 CALL ex06_23WithSignal('Mar2025', 'Morales', 'John', 'XXXXXXX') 51ms
```

The bottom status bar indicates the current file is `ex06_23.sql` and the active line is 113. The bottom right corner shows the system clock as 10:49 PM on 3/5/2025.

Call example 6-23

