Stored procedures

Introduction to PL/SQL:

PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements.All the statements of a block are passed to oracle engine all at once which increases processing speed and decreases the traffic.

PL/SQL stands for “Procedural language extensions to SQL.”

PL/SQL is a database-oriented programming language that extends SQL with procedural capabilities. It was developed by Oracle Corporation within the early 90’s to boost the capabilities of SQL.

PL/SQL adds selective (i.e. if…then…else…) and iterative constructs (ie. loops) to SQL. PL/SQL is most helpful to put in writing triggers and keep procedures. Stored procedures square measure units of procedural code keep during a compiled type inside the info.

Advantages of PL/SQL are as following below:

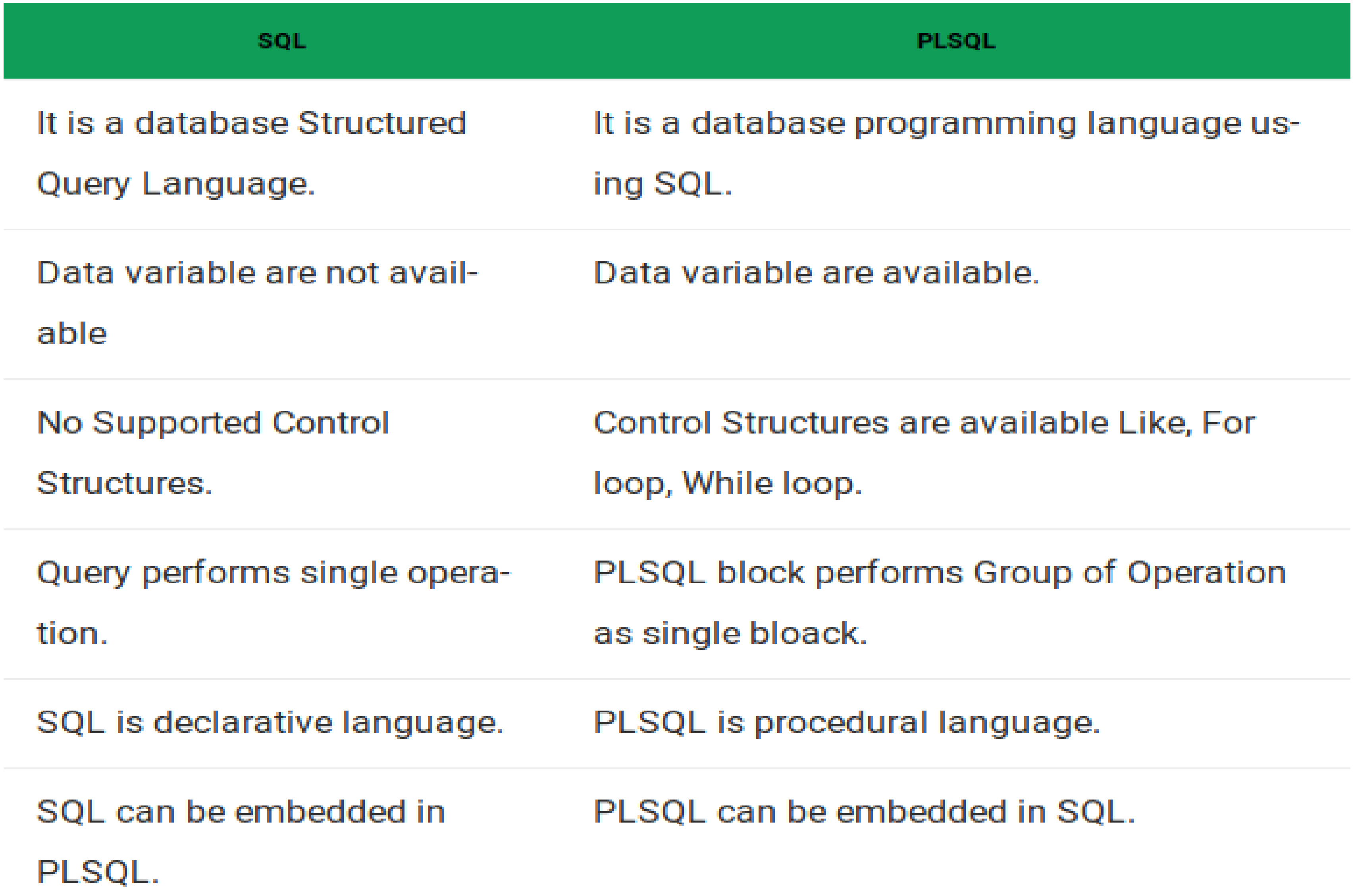
Block structures: It consists of blocks of code, which can be nested within each other. Each block forms a unit of a task or a logical module. PL/SQL blocks are often keep within the info and reused.

Procedural language capability: It consists of procedural language constructs like conditional statements (if else statements) and loops like (FOR loops).

Better performance: PL/SQL engine processes multiple SQL statements at the same time as one block, thereby reducing network traffic.

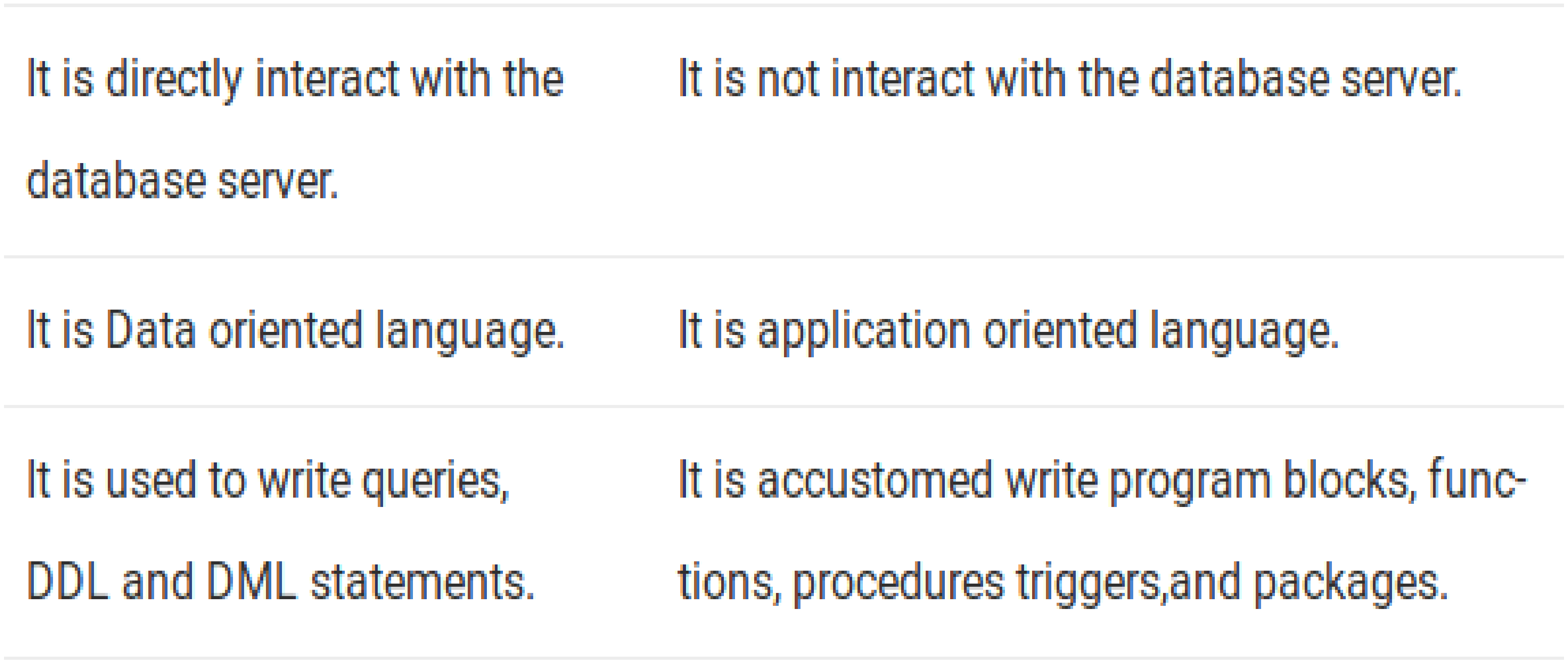
Error handling: PL/SQL handles errors or exceptions effectively throughout the execution of a PL/SQL program.

Comparisons of SQL and PLSQL:



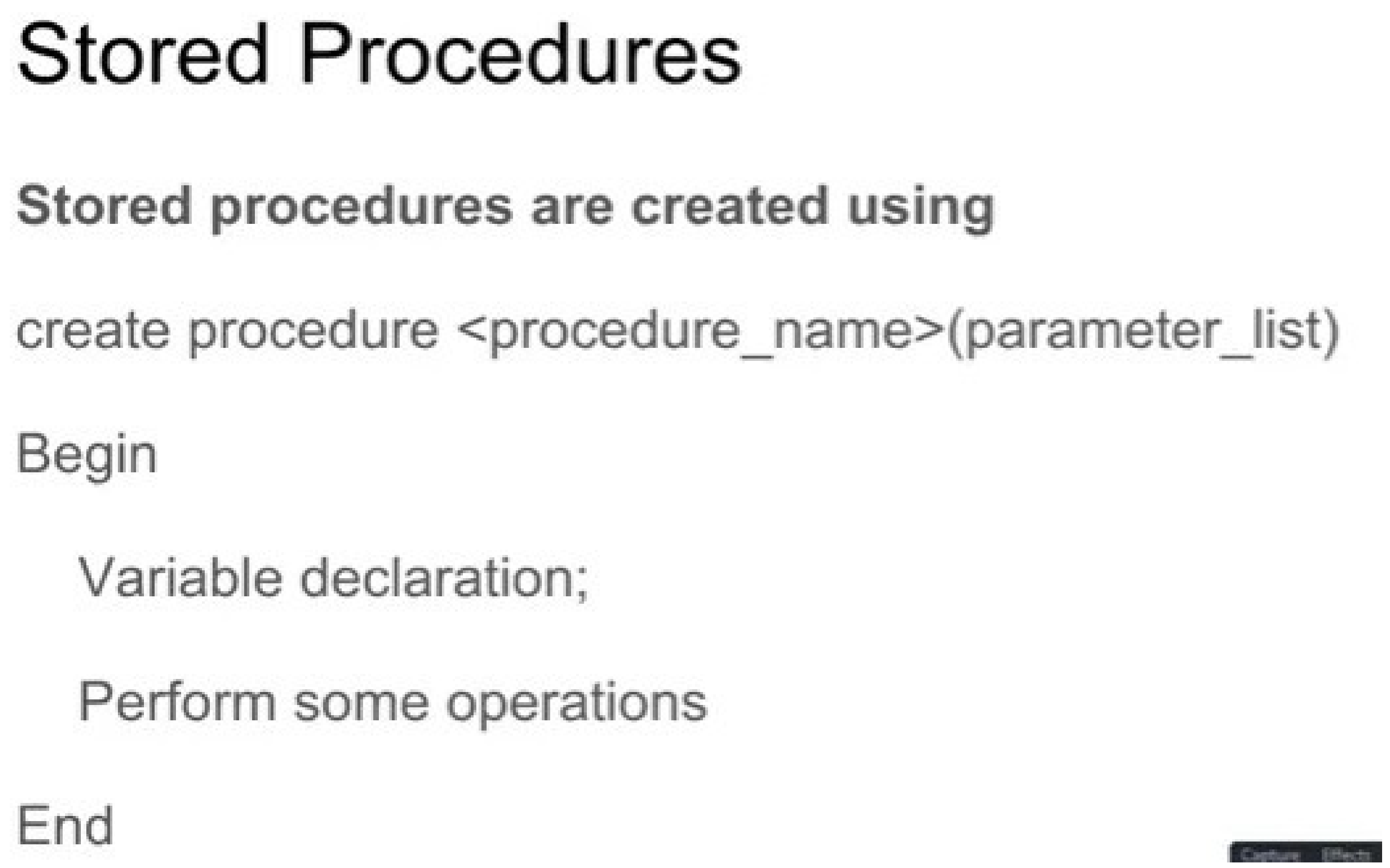
SQL	PLSQL
It is a database Structured Query Language.	It is a database programming language using SQL.
Data variable are not available	Data variable are available.
No Supported Control Structures.	Control Structures are available Like, For loop, While loop.
Query performs single operation.	PLSQL block performs Group of Operation as single bloack.
SQL is declarative language.	PLSQL is procedural language.
SQL can be embedded in PLSQL.	PLSQL can be embedded in SQL.

Comparisons of SQL and PLSQL:



It is directly interact with the database server.	It is not interact with the database server.
It is Data oriented language.	It is application oriented language.
It is used to write queries, DDL and DML statements.	It is accustomed write program blocks, functions, procedures triggers,and packages.

SYNTAX

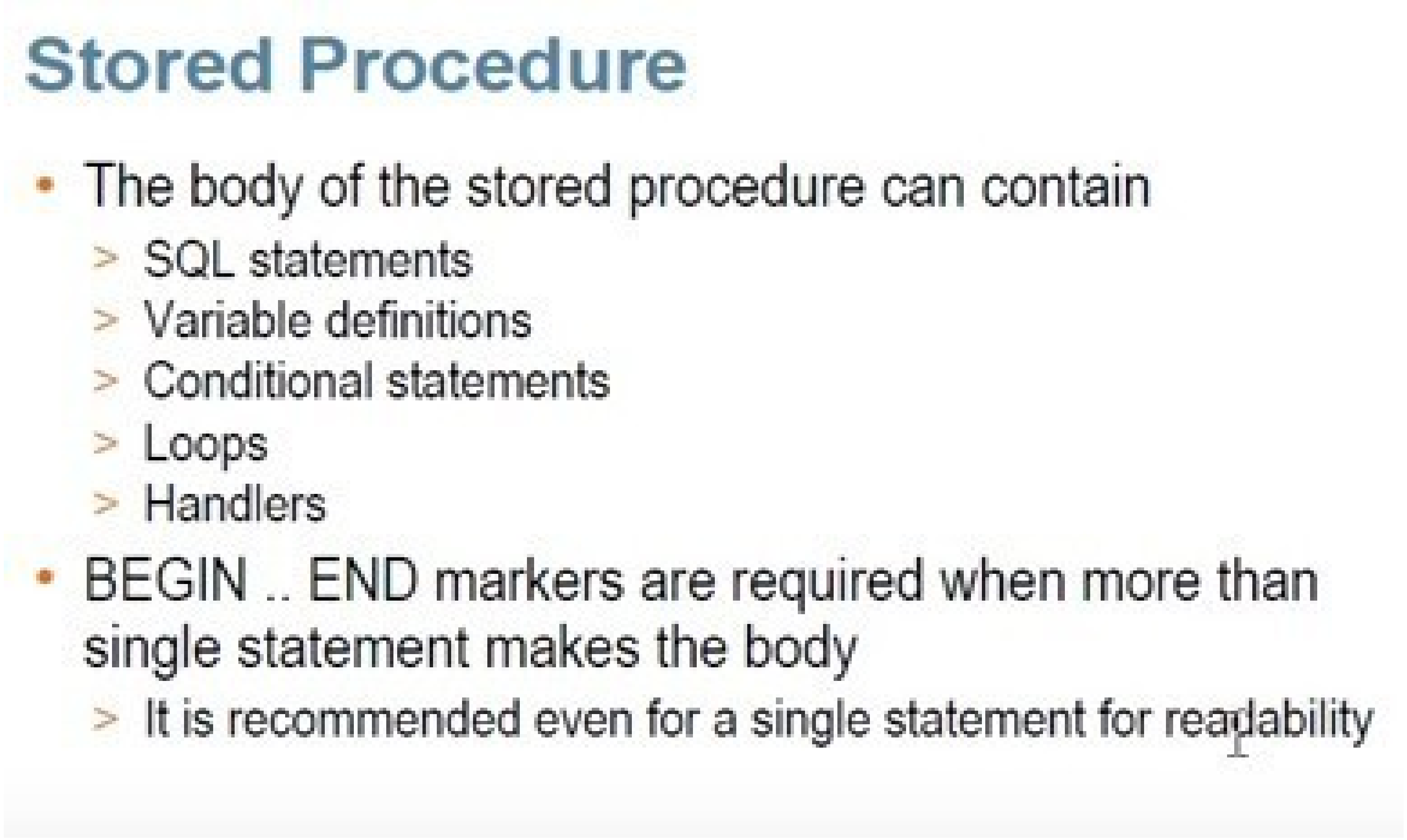


Stored Procedures

Stored procedures are created using

```
create procedure <procedure_name>(parameter_list)
Begin
    Variable declaration;
    Perform some operations
End
```

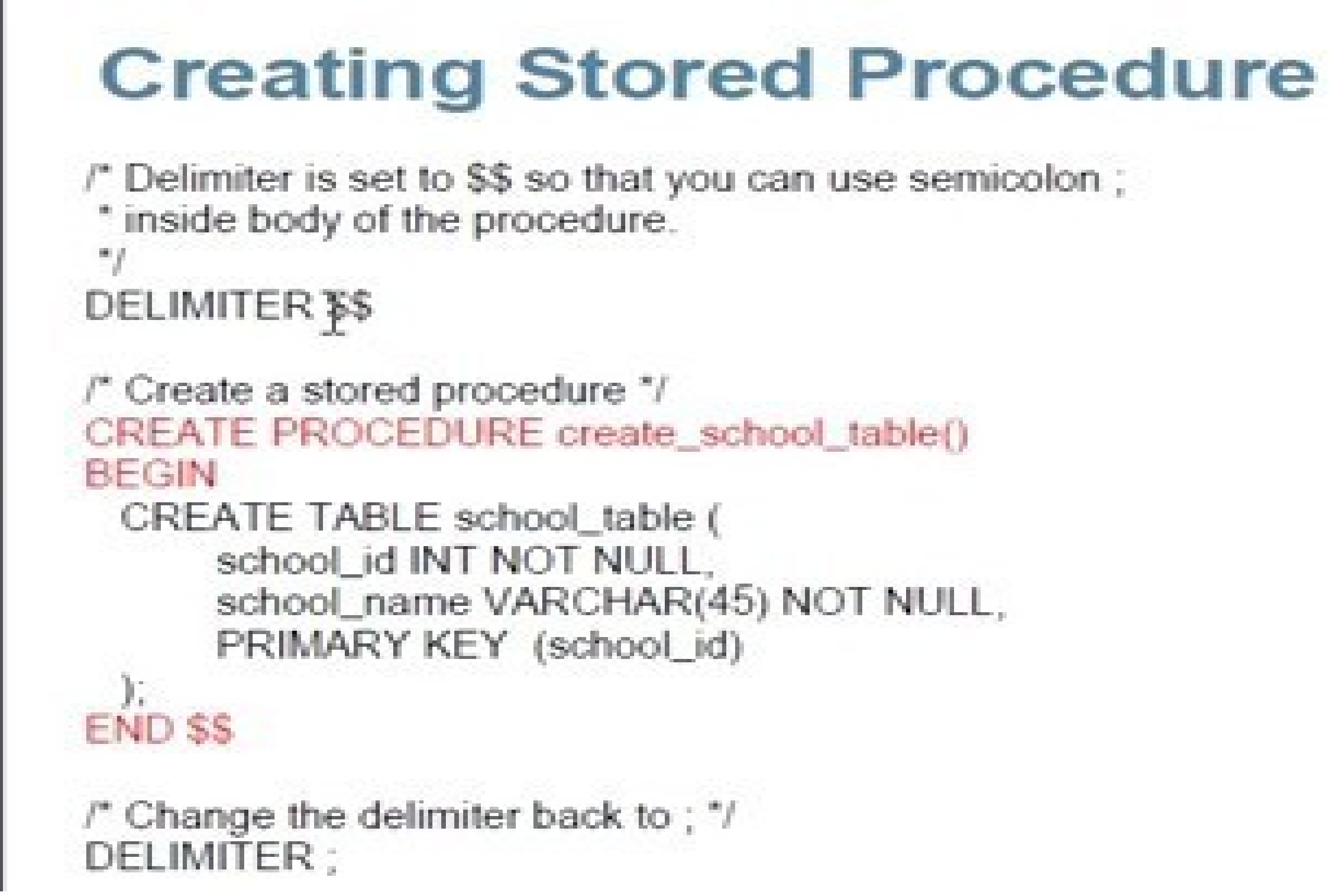
Contd…



Stored Procedure

- The body of the stored procedure can contain
  - SQL statements
  - Variable definitions
  - Conditional statements
  - Loops
  - Handlers
- BEGIN .. END markers are required when more than single statement makes the body
  - It is recommended even for a single statement for readability

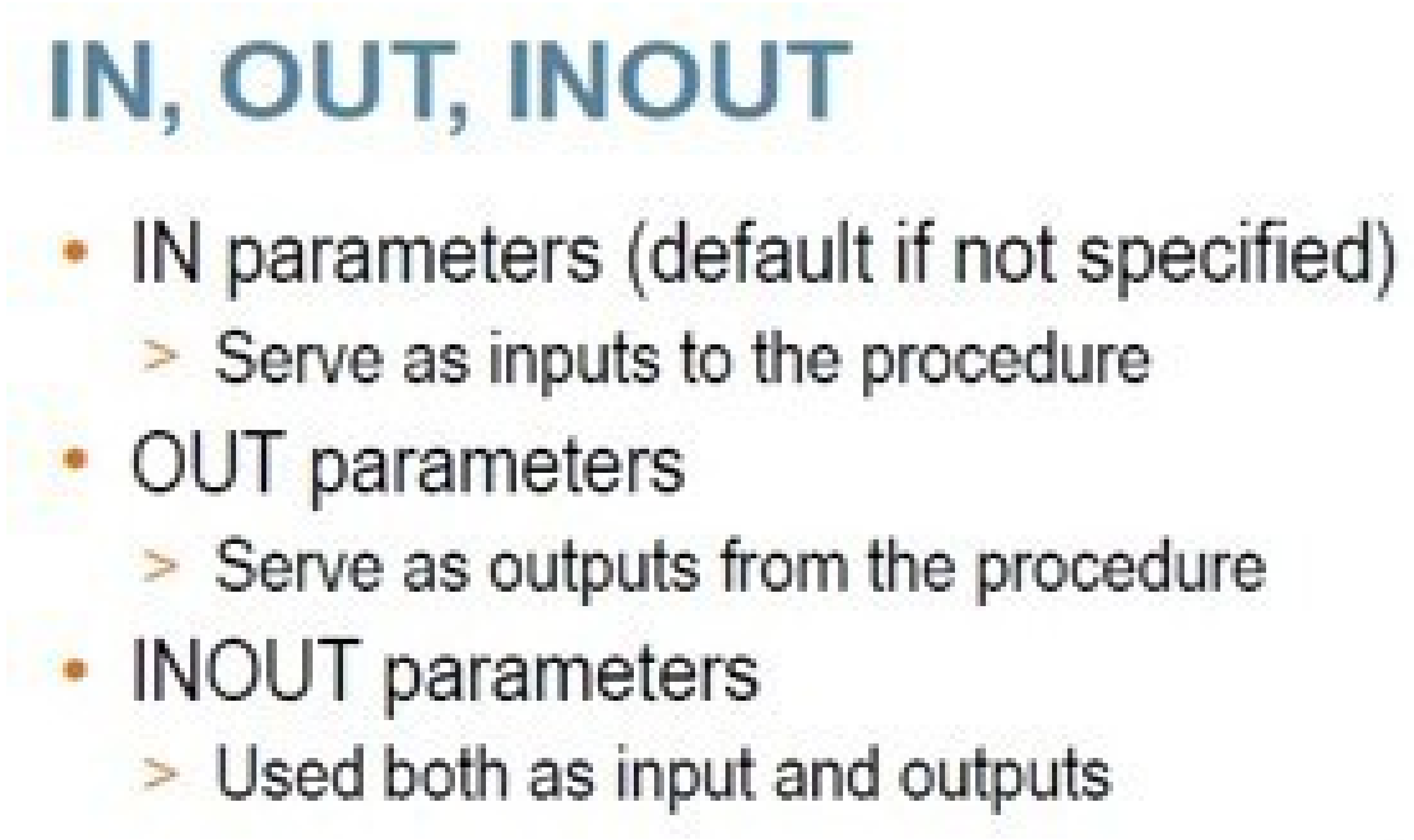
Contd..



Creating Stored Procedure

```
/* Delimiter is set to $$ so that you can use semicolon ;
 * inside body of the procedure.
 */
DELIMITER $$

/* Create a stored procedure */
CREATE PROCEDURE create_school_table()
BEGIN
   CREATE TABLE school_table (
           school_id INT NOT NULL,
           school_name VARCHAR(45) NOT NULL,
           PRIMARY KEY  (school_id)
   );
END $$

/* Change the delimiter back to ; */
DELIMITER ;
```

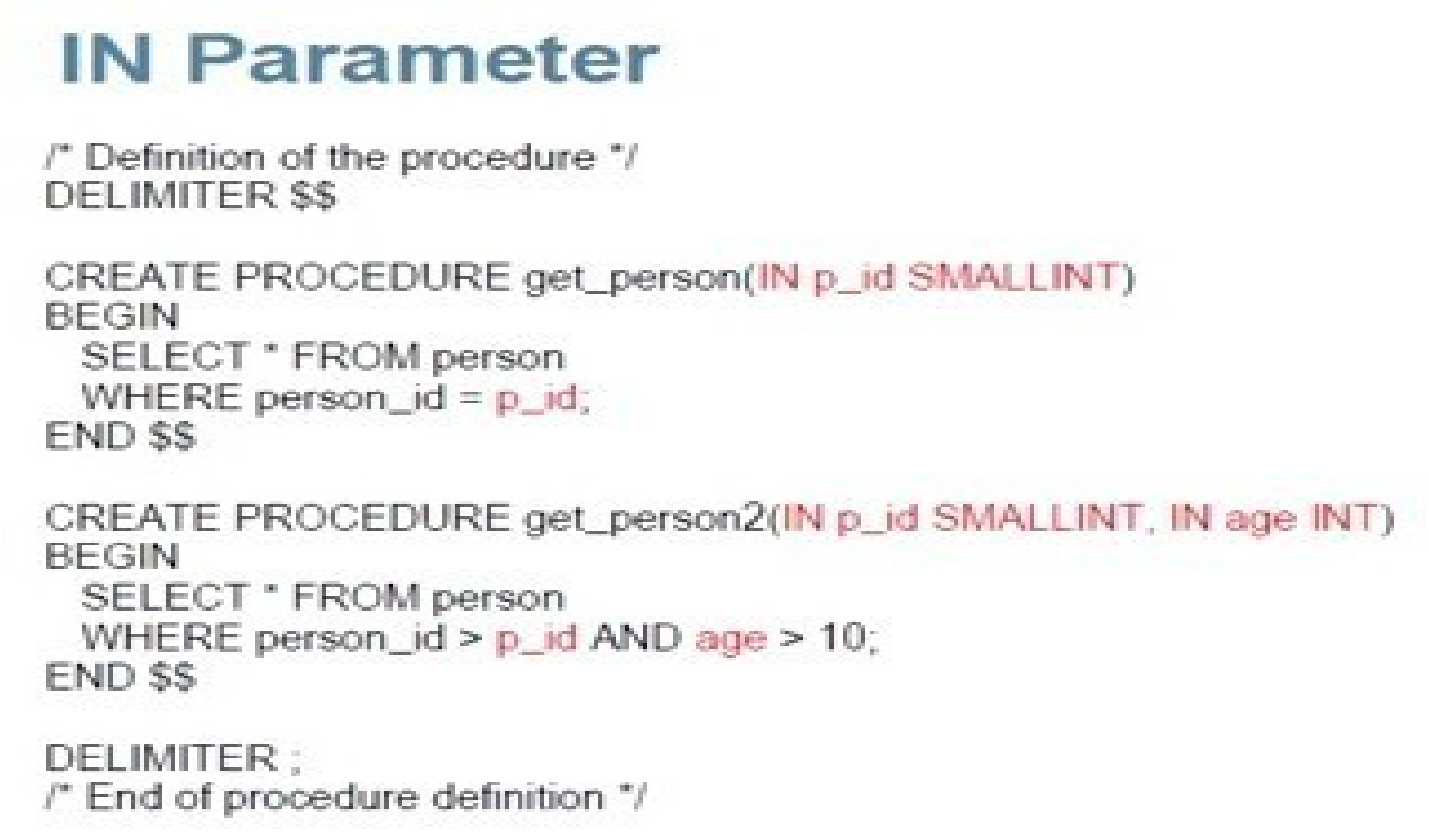
Stored procedure parameters



IN, OUT, INOUT

- IN parameters (default if not specified)
  - > Serve as inputs to the procedure
- OUT parameters
  - > Serve as outputs from the procedure
- INOUT parameters
  - > Used both as input and outputs

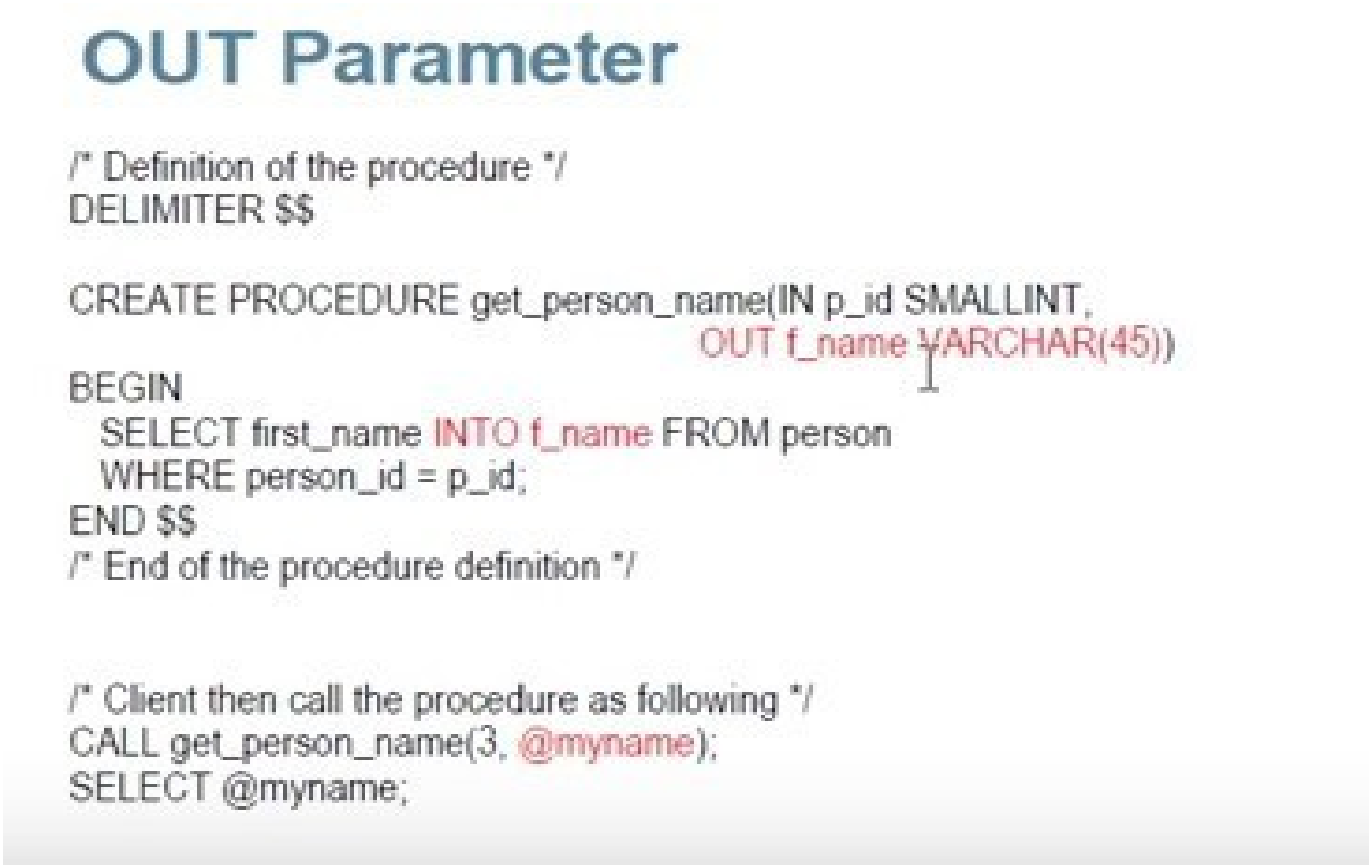
USING IN PARAMETER



IN Parameter

```
/* Definition of the procedure */
DELIMITER $$

CREATE PROCEDURE get_person(IN p_id SMALLINT)
BEGIN
  SELECT * FROM person
  WHERE person_id = p_id;
END $$

CREATE PROCEDURE get_person2(IN p_id SMALLINT, IN age INT)
BEGIN
  SELECT * FROM person
  WHERE person_id > p_id AND age > 10;
END $$

DELIMITER ;
/* End of procedure definition */
```

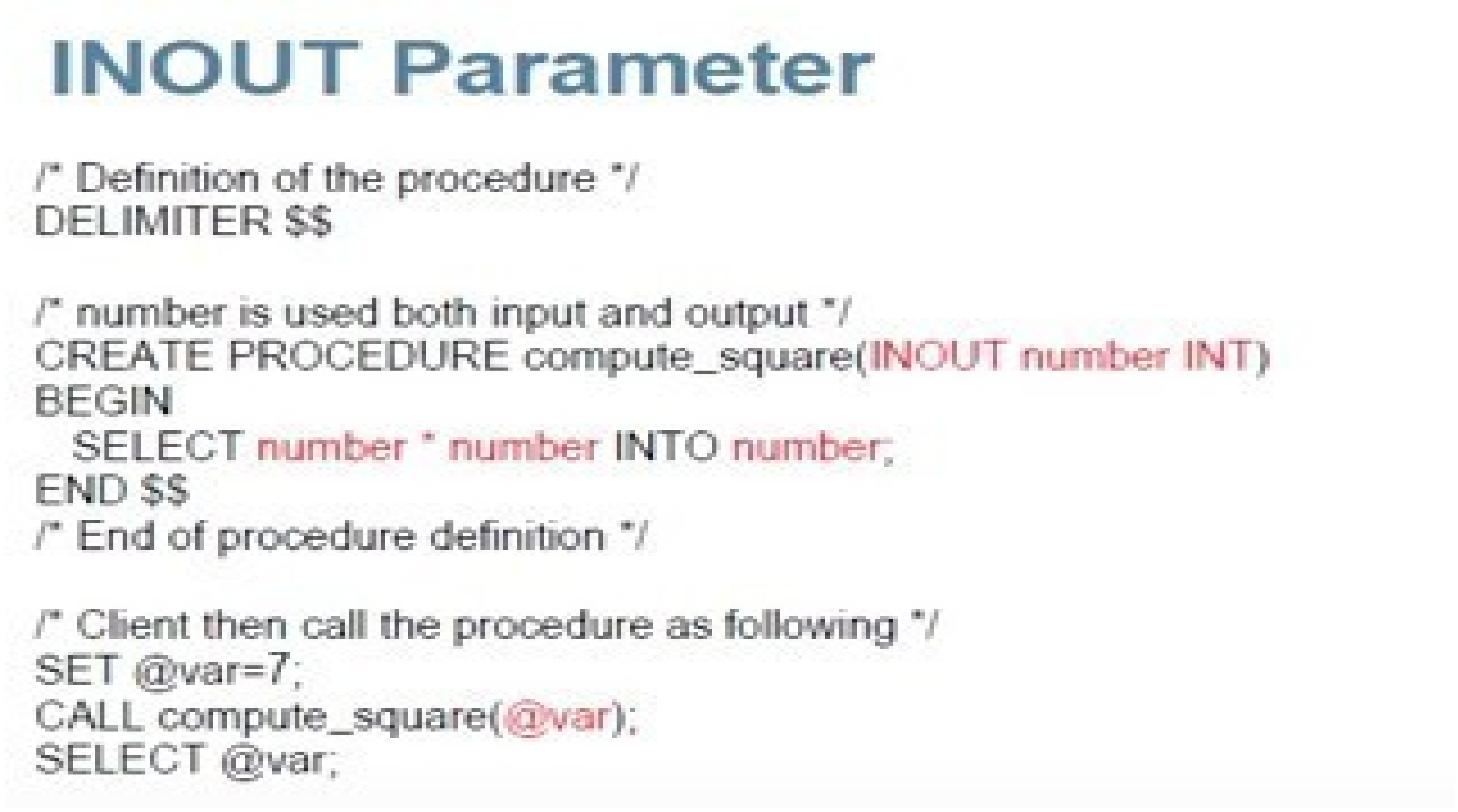
USING OUT PARAMETER



OUT Parameter

```
/* Definition of the procedure */
DELIMITER $$

CREATE PROCEDURE get_person_name(IN p_id SMALLINT,
                                 OUT f_name VARCHAR(45))
BEGIN
   SELECT first_name INTO f_name FROM person
   WHERE person_id = p_id;
END $$
/* End of the procedure definition */


/* Client then call the procedure as following */
CALL get_person_name(3, @myname);
SELECT @myname;
```

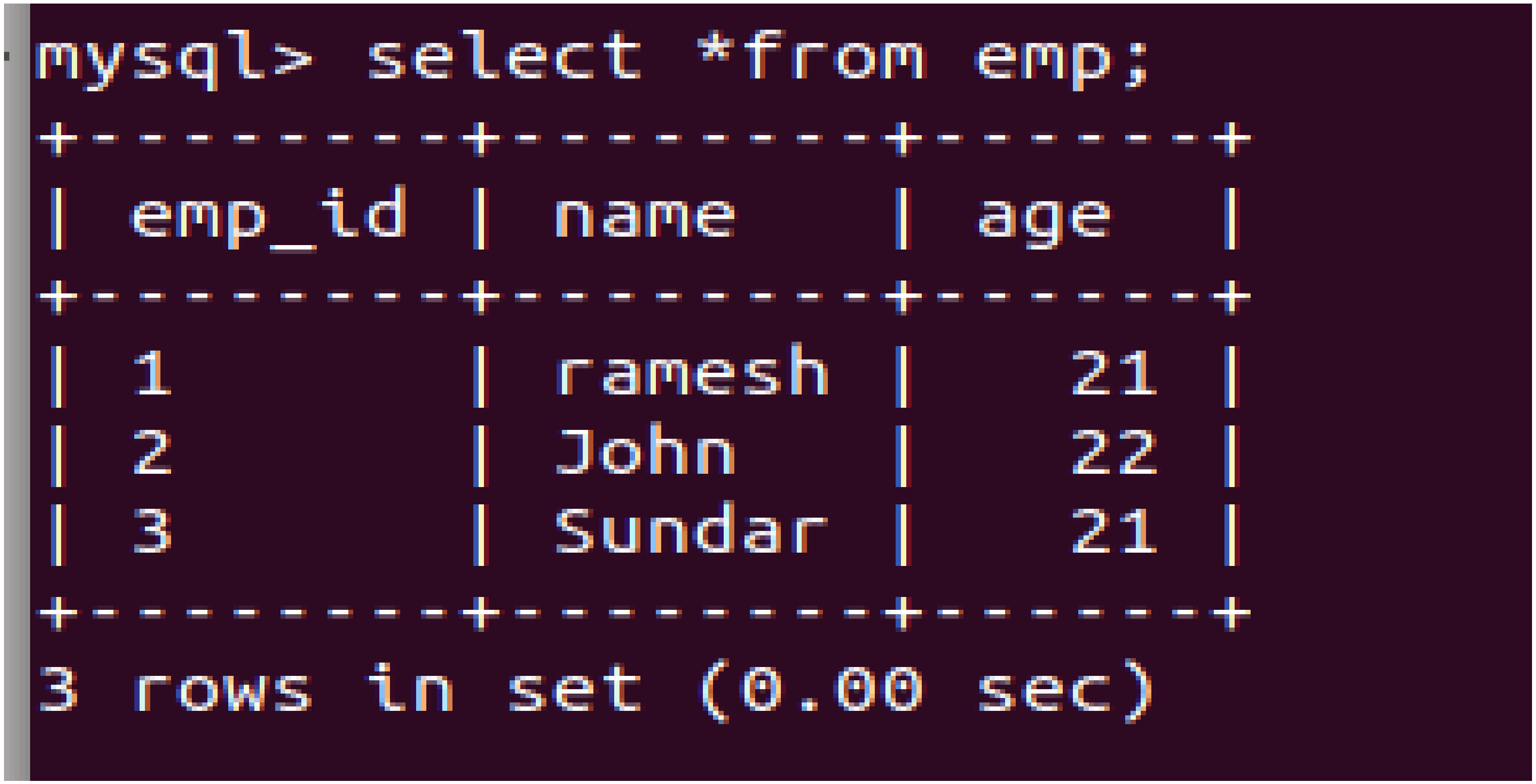
INOUT PARAMETER



INOUT Parameter

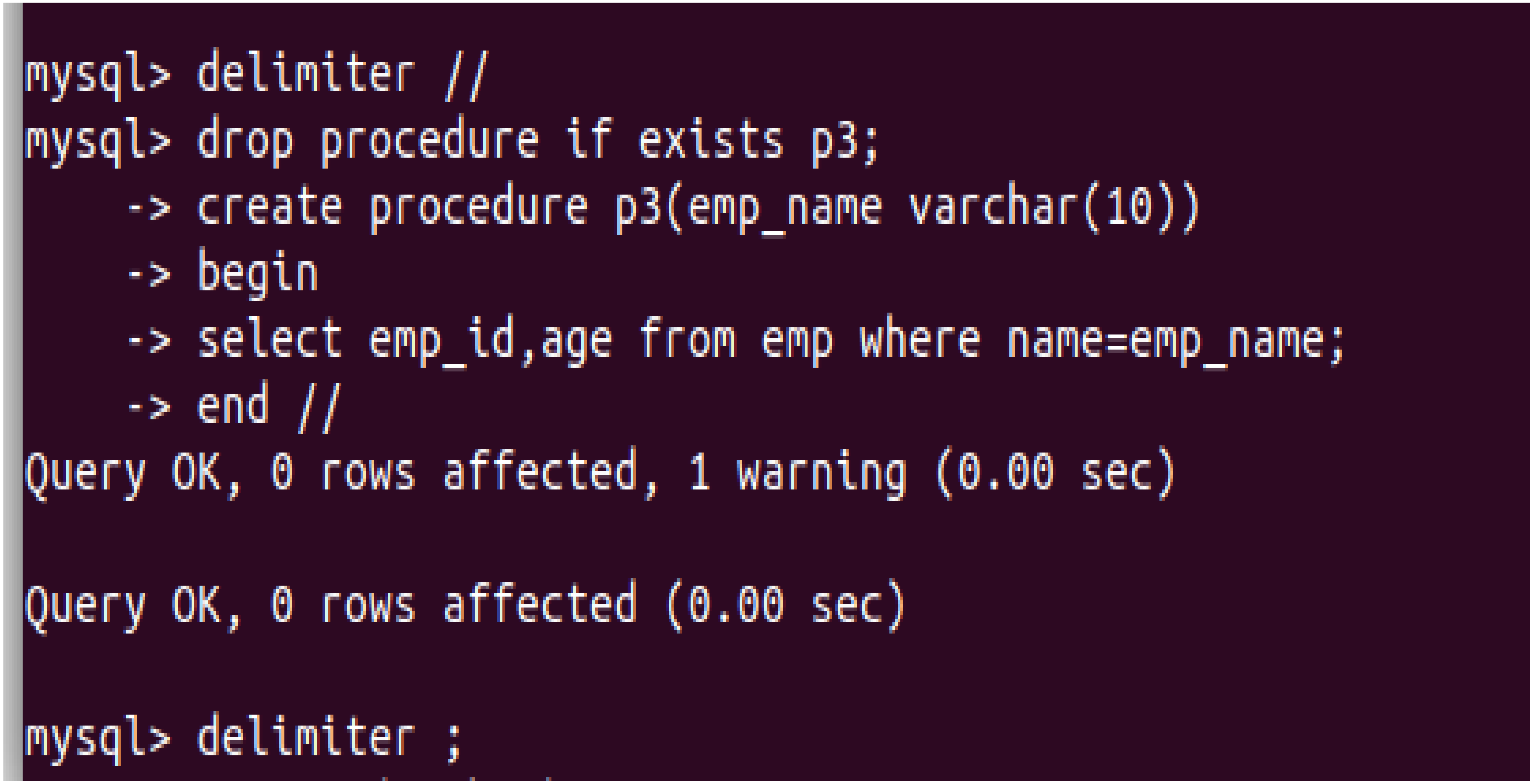
```
/* Definition of the procedure */
DELIMITER $$

/* number is used both input and output */
CREATE PROCEDURE compute_square(INOUT number INT)
BEGIN
  SELECT number * number INTO number;
END $$
/* End of procedure definition */

/* Client then call the procedure as following */
SET @var=7;
CALL compute_square(@var);
SELECT @var;
```

Stored Procedure



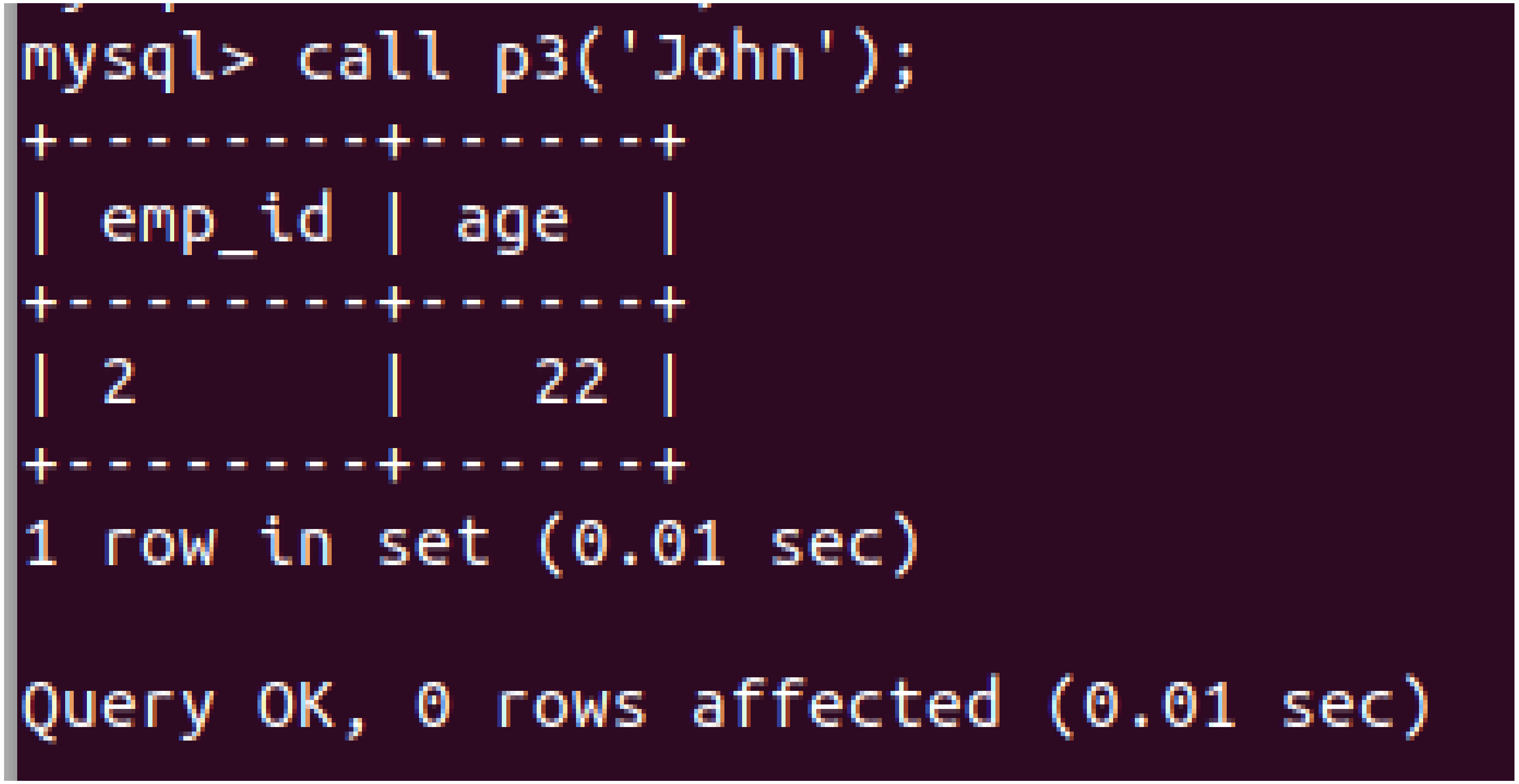
```
mysql> select *from emp;
+--------+---------+-------+
| emp_id | name    | age   |
+--------+---------+-------+
| 1      | ramesh  |    21 |
| 2      | John    |    22 |
| 3      | Sundar  |    21 |
+--------+---------+-------+
3 rows in set (0.00 sec)
```

Contd…



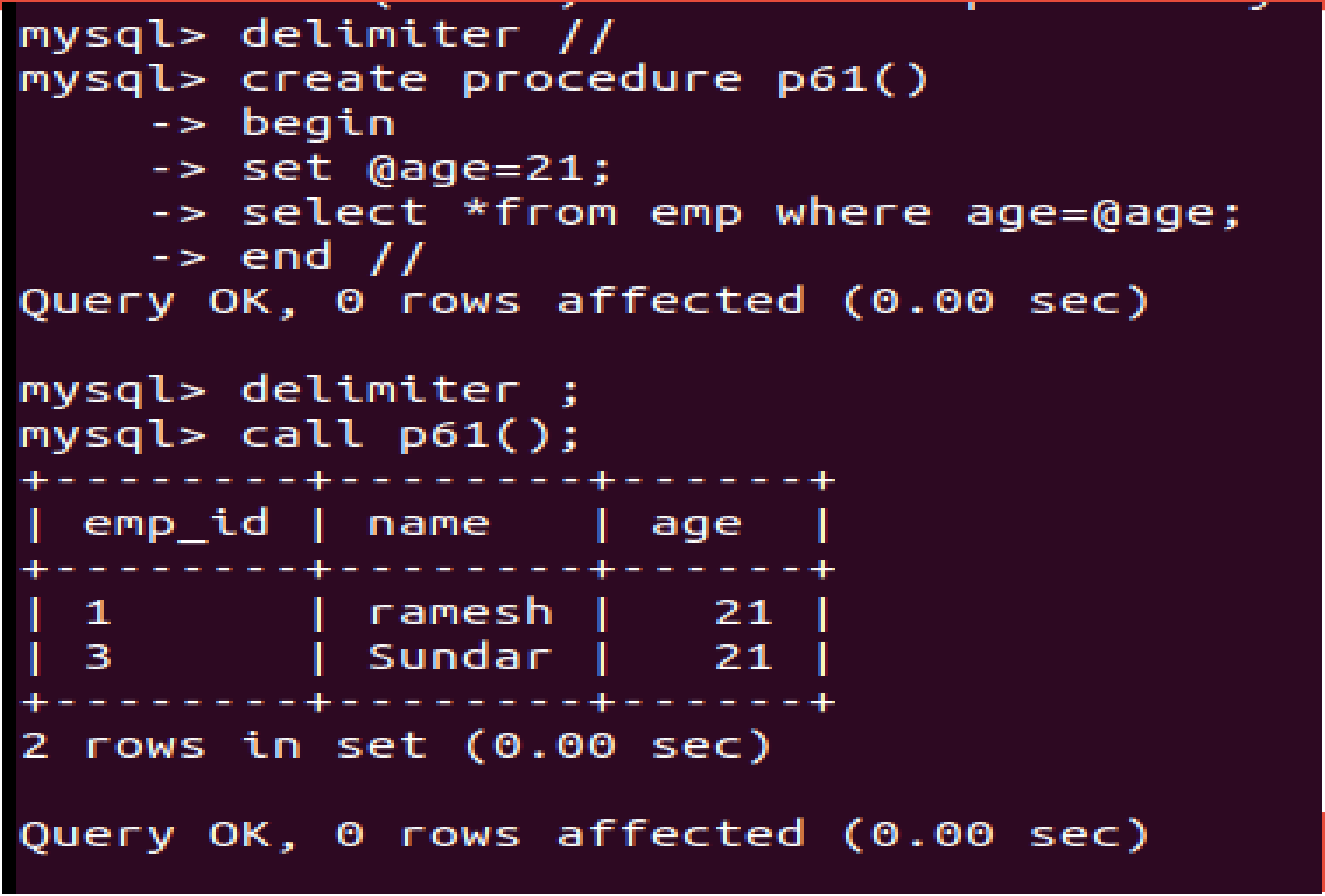
```
mysql> delimiter //
mysql> drop procedure if exists p3;
    -> create procedure p3(emp_name varchar(10))
    -> begin
    -> select emp_id,age from emp where name=emp_name;
    -> end //
Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

Contd...



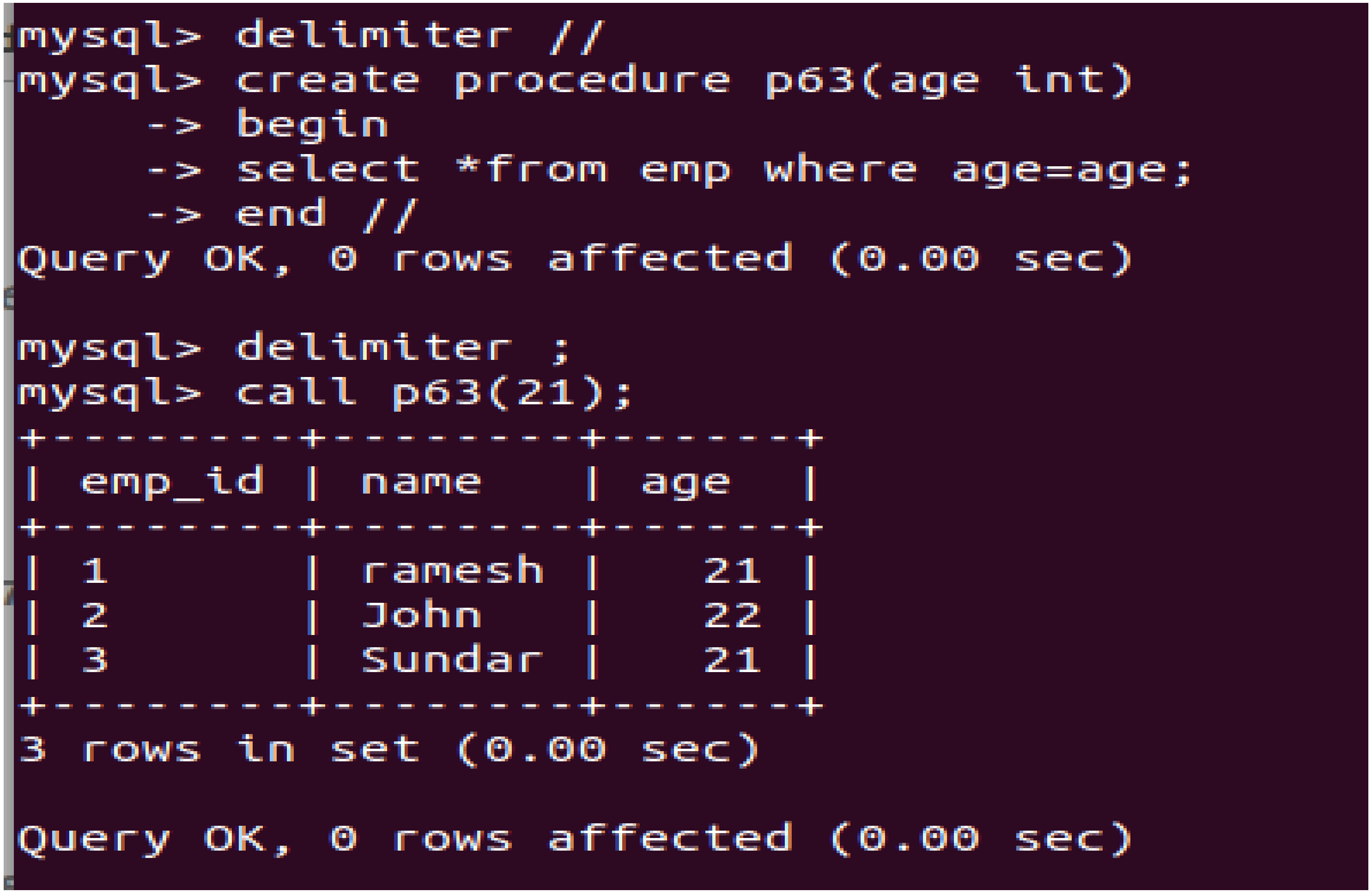
```
mysql> call p3('John');
+---------+-------+
| emp_id | age   |
+---------+-------+
| 2       |    22 |
+---------+-------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```

USING SET



```
mysql> delimiter //
mysql> create procedure p61()
    -> begin
    -> set @age=21;
    -> select *from emp where age=@age;
    -> end //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
mysql> call p61();
+--------+---------+-------+
| emp_id | name    | age   |
+--------+---------+-------+
| 1      | ramesh  |    21 |
| 3      | Sundar  |    21 |
+--------+---------+-------+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

Without using SET



```
mysql> delimiter //
mysql> create procedure p63(age int)
    -> begin
    -> select *from emp where age=age;
    -> end //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
mysql> call p63(21);
+--------+---------+------+
| emp_id | name    | age  |
+--------+---------+------+
| 1      | ramesh  |   21 |
| 2      | John    |   22 |
| 3      | Sundar  |   21 |
+--------+---------+------+
3 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

Benefits of Stored Procedures

Modular Programming – You can write a stored procedure once, then call it from multiple places in your application.

Performance - Stored procedures provide faster code execution and reduce network traffic.

<u>Faster execution</u>: Stored procedures are parsed and optimized as soon as they are created and the stored procedure is stored in memory.

<u>Reduced network traffic</u>: If you send many lines of SQL code over the network to your SQL Server, this will impact on network performance. This is especially true if you have hundreds of lines of SQL code and/or you have lots of activity on your application. Running the code on the SQL Server (as a stored procedure) eliminates the need to send this code over the network. The only network traffic will be the parameters supplied and the results of any query.

Security - Users can execute a stored procedure without needing to execute any of the statements directly. Therefore, a stored procedure can provide advanced database functionality for users who wouldn't normally have access to these tasks, but this functionality is made available in a tightly controlled way.

Drawbacks of Stored Procedures

- It’s debugging is hard
- It is not database independent. Its code may vary based on database server