Stored Procedures and Triggers - I

🎯 Session Objectives

- ✅ Understand the concept of stored procedures.
- ✅ Differentiate between IN, OUT, and INOUT parameters.
- ✅ Create and use procedures with real-world examples.

DRY

[Don't Repeat Yourself]

SYNTAX:

```
DELIMITER $$

CREATE PROCEDURE procedure_name (
  [IN | OUT | INOUT] param_name DATATYPE,
  ...
)
BEGIN
  -- Declaration section (optional)
  -- Executable SQL statements
END $$

DELIMITER ;
```

Functions - User Defined / Pre-Defined
Methods() -> invoked from library

python

packages

.strip()
.split()
.swapcase()
.isnumeric()

```
CREATE TABLE student_info (
  stud_id INT PRIMARY KEY,
  stud_code INT,
  stud_name VARCHAR(50),
  subject VARCHAR(50),
  marks INT,
  phone VARCHAR(15)
);
```

```
INSERT INTO student_info VALUES
(1, 101, 'Mark', 'English', 68, '3454569357'),
(2, 102, 'Joseph', 'Physics', 70, '9876543659'),
(3, 103, 'John', 'Maths', 70, '9765326975'),
(4, 104, 'Barack', 'Maths', 92, '87069873256'),
(5, 105, 'Rinky', 'Maths', 85, '6753159757'),
(6, 106, 'Adam', 'Science', 82, '79642256864'),
(7, 107, 'Andrew', 'Science', 83, '5674243579'),
(8, 108, 'Brayan', 'Science', 83, '7524316576'),
(9, 109, 'Alexandar', 'Biology', 67, '2347346438');
```

stud_id	stud_code	stud_name	subject	marks	phone
1	101	Mark	English	68	3454569357
2	102	Joseph	Physics	70	9876543659
3	103	John	Maths	70	9765326975
4	104	Barack	Maths	92	87069873256
5	105	Rinky	Maths	85	6753159757
6	106	Adam	Science	82	79642256864
7	107	Andrew	Science	83	5674243579
8	108	Brayan	Science	83	7524316576
9	109	Alexandar	Biology	67	2347346438

Field	Type	Null	Key	Default	Extra
stud_id	int	NO	PRI	NULL	
stud_code	int	YES		NULL	
stud_name	varchar...	YES		NULL	
subject	varchar...	YES		NULL	
marks	int	YES		NULL	
phone	varchar...	YES		NULL	


```
DELIMITER $$
CREATE PROCEDURE get_merit_student()
BEGIN
    SELECT * FROM student_info WHERE marks > 70;
    SELECT COUNT(*) AS total_student FROM student_info;
END $$
DELIMITER ;
```

```
CALL get_merit_student();
```

stud_id	stud_code	stud_name	subject	marks	phone
4	104	Barack	Maths	92	87069873256
5	105	Rinky	Maths	85	6753159757
6	106	Adam	Science	82	79642256864
7	107	Andrew	Science	83	5674243579
8	108	Brayan	Science	83	7524316576
NULL	NULL	NULL	NULL	NULL	NULL

total_student
9

```
877 •  SHOW Procedure STATUS;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Db	Name	Type	Definer	Modified	Created	Security_type
bike_analysis	get_merit_student	PROCEDURE	root@localhost	2025-07-12 21:29:04	2025-07-12 21:29:04	DEFINER

```
878 •  SHOW Procedure STATUS WHERE Db = "bike_analysis";
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Db	Name	Type	Definer	Modified	Created	Security_type	Comment	character_set_client	collation_connection	Database Collation
bike_analysis	get_merit_student	PROCEDURE	root@localhost	2025-07-12 21:29:04	2025-07-12 21:29:04	DEFINER		utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

```
-- Managing Store Procedure
SHOW Procedure STATUS;
SHOW Procedure STATUS WHERE Db = "bike_analysis";

-- There is no option to alter the procedure - Just Drop and Recreate it.
DROP PROCEDURE get_merit_student; -- just tell the name of the procedure

DELIMITER $$
CREATE PROCEDURE get_merit_student()
BEGIN
    SELECT * FROM student_info WHERE marks > 70;
   -- SELECT COUNT(*) AS total_student FROM student_info;
END $$
DELIMITER ;

CALL get_merit_student();
```

stud_id	stud_code	stud_name	subject	marks	phone
4	104	Barack	Maths	92	87069873256
5	105	Rinky	Maths	85	6753159757
6	106	Adam	Science	82	79642256864
7	107	Andrew	Science	83	5674243579
8	108	Brayan	Science	83	7524316576

It shows all the students with marks > 70.
What if I'll insert more data, and recall the procedure? Do we get the updated data?

```
893 •  INSERT INTO student_info VALUES
894    (10, 101, 'Moon', 'English', 78, '3454569357'),
895    (11, 102, 'Josephic', 'Physics', 80, '9876543659'),
896    (12, 103, 'Johny', 'Maths', 90, '9765326975');
897 •  SELECT * FROM student_info;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

stud_id	stud_code	stud_name	subject	marks	phone
1	101	Mark	English	68	3454569357
2	102	Joseph	Physics	70	9876543659
3	103	John	Maths	70	9765326975
4	104	Barack	Maths	92	87069873256
5	105	Rinky	Maths	85	6753159757
6	106	Adam	Science	82	79642256864
7	107	Andrew	Science	83	5674243579
8	108	Brayan	Science	83	7524316576
9	109	Alexandar	Biology	67	2347346438
10	101	Moon	English	78	3454569357
11	102	Josephic	Physics	80	9876543659
12	103	Johny	Maths	90	9765326975
NULL	NULL	NULL	NULL	NULL	NULL

```
CALL get_merit_student();
```

stud_id	stud_code	stud_name	subject	marks	phone
4	104	Barack	Maths	92	87069873256
5	105	Rinky	Maths	85	6753159757
6	106	Adam	Science	82	79642256864
7	107	Andrew	Science	83	5674243579
8	108	Brayan	Science	83	7524316576
10	101	Moon	English	78	3454569357
11	102	Josephic	Physics	80	9876543659
12	103	Johny	Maths	90	9765326975

IN Parameter -> Accepting the Input

Limit the numbers of records by taking input var -> Limit value.

```
901    DELIMITER $$
902 •  CREATE PROCEDURE get_student(IN var1 INT)
903    BEGIN
904        SELECT * FROM student_info LIMIT var1;
905        -- SELECT COUNT(*) AS total_student FROM student_info;
906    END $$
907    DELIMITER ;
908
909 •  CALL get_student(5);
```

INPUT

Memory

var1 = NULL → 5

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

stud_id	stud_code	stud_name	subject	marks	phone
1	101	Mark	English	68	3454569357
2	102	Joseph	Physics	70	9876543659
3	103	John	Maths	70	9765326975
4	104	Barack	Maths	92	87069873256
5	105	Rinky	Maths	85	6753159757

```
909 •  CALL get_student(7);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

stud_id	stud_code	stud_name	subject	marks	phone
1	101	Mark	English	68	3454569357
2	102	Joseph	Physics	70	9876543659
3	103	John	Maths	70	9765326975
4	104	Barack	Maths	92	87069873256
5	105	Rinky	Maths	85	6753159757
6	106	Adam	Science	82	79642256864
7	107	Andrew	Science	83	5674243579

```
CALL get_student();
-- Error Code: 1318. Incorrect number of arguments for PROCEDURE
bike_analysis.get_student; expected 1, got 0
```

Let's take input variable as subject_name to apply filter.

```
DELIMITER $$
CREATE PROCEDURE get_students_by_subject(IN sub_name VARCHAR(20))
BEGIN
    SELECT stud_name , marks , subject
    FROM student_info
    WHERE subject = sub_name;
END $$
DELIMITER ;

CALL get_students_by_subject("Maths");
```

stud_name	marks	subject
John	70	Maths
Barack	92	Maths
Rinky	85	Maths
Johny	90	Maths

```
924 •  CALL get_students_by_subject("Science");
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

stud_name	marks	subject
Adam	82	Science
Andrew	83	Science
Brayan	83	Science

OUT PARAMETER:

Display the highest Marks from student_info;

```
-- OUT PARAMETER
DELIMITER $$
CREATE PROCEDURE display_highest_marks(OUT highestmarks INT)
BEGIN
        SELECT MAX(marks) INTO highestmarks FROM student_info;
END $$
DELIMITER ;

CALL display_highest_marks(@marks);
SELECT @marks;
```

@marks
92

Memory

highestmarks = null → 92

@marks = null → 92

```
934 •  CALL display_highest_marks(@m);
935 •  SELECT @m;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Cor

@m
92

stud_id	stud_code	stud_name	subject	marks	phone
1	101	Mark	English	68	3454569357
2	102	Joseph	Physics	70	9876543659
3	103	John	Maths	70	9765326975
4	104	Barack	Maths	92	87069873256
5	105	Rinky	Maths	85	6753159757
6	106	Adam	Science	82	79642256864
7	107	Andrew	Science	83	5674243579
8	108	Brayan	Science	83	7524316576
9	109	Alexandar	Biology	67	2347346438
10	101	Moon	English	78	3454569357
11	102	Josephic	Physics	80	9876543659
12	103	Johny	Maths	90	9765326975
NULL	NULL	NULL	NULL	NULL	NULL

max(marks) = -Inf → 68 → 70 → 92

IN - OUT Parameter

Count the number of students by subject

subject - IN
total_count - OUT

```
DELIMITER $$
CREATE PROCEDURE count_students_by_subject(IN sub_name VARCHAR(20), OUT total_count INT)
BEGIN
    SELECT count(*) INTO total_count FROM student_info WHERE subject = sub_name;
END $$
DELIMITER ;

CALL count_students_by_subject('Science', @student_count);
SELECT @student_count;
```

@student_count
3

```
949 •  CALL count_students_by_subject('Maths', @student_count);
950 •  SELECT @student_count;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

@student_count
4

Calculate the average marks in a subject

subject - IN
avg_marks - OUT

```
DELIMITER $$
CREATE PROCEDURE get_avg_marks(IN sub_name VARCHAR(20), OUT avg_marks DECIMAL(5,2))
BEGIN
    SELECT
        AVG(marks)  INTO avg_marks
    FROM student_info WHERE subject = sub_name;
END $$
DELIMITER ;

CALL get_avg_marks('Maths', @avg_marks);
SELECT @avg_marks;
```

@avg_marks
84.25

```
970 •  SELECT
971        AVG(marks)
972    FROM student_info
973    WHERE subject = "Maths";
```

Result Grid | Filter Rows: | Export:

AVG(marks)
84.2500

INOUT Parameter

```
-- INOUT Parameter

DELIMITER $$
CREATE PROCEDURE display_marks(INOUT var1 INT)
BEGIN
        SELECT marks INTO var1
    FROM student_info WHERE stud_id = var1;
END $$
DELIMITER ;
SET @score = 3;
CALL display_marks(@score);
SELECT @score;
```

Memory

```
@score = 3  70
var1 = null
       3  70
```

@score
70

stud_id	stud_code	stud_name	subject	marks	phone
1	101	Mark	English	68	3454569357
2	102	Joseph	Physics	70	9876543659
3	103	John	Maths	70	9765326975
4	104	Barack	Maths	92	87069873256
5	105	Rinky	Maths	85	6753159757
6	106	Adam	Science	82	79642256864
7	107	Andrew	Science	83	5674243579
8	108	Brayan	Science	83	7524316576
9	109	Alexandar	Biology	67	2347346438
10	101	Moon	English	78	3454569357
11	102	Josephic	Physics	80	9876543659
12	103	Johny	Maths	90	9765326975
NULL	NULL	NULL	NULL	NULL	NULL