MySQL Procedures
(LAB Lecture)

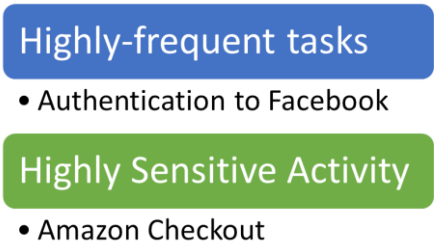
Department of Information Technology
Database Systems II (IT216)
Spring 2025-2026
Week 10 – April 7, 2026
Lecturer: Soma Soleimanzadeh

Contents

- Stored Routines in MySQL
  - Functions
  - Procedures
- Procedures
- Syntax of Creating Procedure in MySQL

Procedures in MySQL

- **Procedure** is a block of code bundled under a name that can be reused for two types of tasks:



- **Procedures** can be reused over and over again.
- Usually **INSERT**, **UPDATE** or **DELETE**.

3

Syntax of Creating Procedure in MySQL

```
DELIMITER   %%

CREATE  PROCEDURE  procedure_name (IN / OUT  parameter1  datatype1,
                                   IN / OUT  parameter2  datatype2,
                                   … )
BEGIN
        <Declaration Part of Procedure>
        <Execution Part of Procedure>
END%%

DELIMITER   ;
```

4

Procedure

```
DELIMITER //

CREATE PROCEDURE procedure_name
(IN / OUT parameter1 datatype1,
IN / OUT parameter2 datatype2,
... )
BEGIN
        <Declaration Part of Procedure>
        <Execution Part of Procedure>
END//

DELIMITER ;
```

Function

```
DELIMITER //

CREATE FUNCTION function_name
(input_parameter1 datatype1,
 input_parameter2 datatype2,
... )
RETURNS return_datatype
BEGIN
        <Declaration Part of Function>
        <Execution Part of Function>
        RETURN return_value;
END//

DELIMITER ;
```

5

Let's Create the Database and Tables

- Create a database named **hospital** and activate it.
- Create the **Patient** table and enter the data inside the table.
    - **PID** is **primary key** and **auto_increment**.

Patient

PID	PName	Age	Payment
1	Mike	26	300
2	Kate	64	250
3	John	29	1500

```
create database hospital;
use hospital;

create table patient
(PID int auto_increment,
PName varchar(100),
Age int,
payment int,
primary key(PID));
```

```
insert into patient(PName,Age,payment) values ('Mike',26,300),
                                              ('Kate',64,250),
                                              ('John',29,1500);
```

6

Let's Create Some Procedures!

- Create a procedure named **YoungPatients** to display all information of patients younger than 30 years old.
- Invoke (call) the *procedure.*

Patient

PID	PName	Age	Payment
1	Mike	26	300
2	Kate	64	250
3	John	29	1500

```
delimiter %%
create procedure YoungPatients()
    begin
        select * from patient where Age<30;
    end%%
delimiter ;
```

Step 1 – Creating Function

```
call YoungPatients();
```

Step 2 – Calling Function

PID	PName	Age	payment
1	Mike	26	200
3	John	29	1300

Let's Create Some Procedures!

- Create a procedure named **ApplyDiscount** that **takes ID of a patient**, and updates his/her **payment** amount by applying $100 discount.
- Call the procedure for a patient with PID=3.

Patient

PID	PName	Age	Payment
1	Mike	26	300
2	Kate	64	250
3	John	29	1500

```
delimiter $$
create procedure ApplyDiscount(IN ID int)
    begin
        update patient
        set payment = payment - 100
        where PID = ID;
    end$$
delimiter ;
```

Step 1 – Creating Function

```
call ApplyDiscount(3);

select * from patient;
```

Step 2 – Calling Function

PID	PName	Age	payment
1	Mike	26	300
2	Kate	64	250
3	John	29	1400

8

Let's Create Some Procedures!

- Create a procedure named **ageAVG** that **calculates and displays the average age of all patients**.
- Call the procedure.

Patient

PID	PName	Age	Payment
1	Mike	26	300
2	Kate	64	250
3	John	29	1500

```
delimiter &&
create procedure ageAVG(OUT averageAge decimal(3,1))
    begin
        select avg(age) into averageAge from patient;
    end&&
delimiter ;
```

Step 1 – Creating Function

```
set @result=0;
call ageAVG(@result);
select @result;
```

Step 2 – Calling Function

@result
39.7

9

Let's Create Some Procedures!

- Create a procedure named **ageAvgMinMax** that displays the **average age**, **minimum age** and **maximum age** of all patients.
- Call the procedure.

Patient

PID	PName	Age	Payment
1	Mike	26	300
2	Kate	64	250
3	John	29	1500

```
delimiter &&
create procedure ageAvgMinMax(OUT averageAge decimal(3,1),
                              OUT minAge int,
                              OUT maxAge int)
    begin
        select avg(age),min(age),max(age)
        into averageAge, minAge,maxAge
        from patient;
    end&&
delimiter ;
```

Step 1 – Creating Function

```
set @average = 0;
set @minimum = 0;
set @maximum = 0;
call ageAvgMinMax(@average, @minimum,@maximum);
select @average, @minimum, @maximum;
```

Step 2 – Calling Function

@average	@minimum	@maximum
39.7	26	64

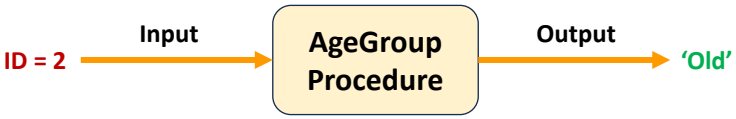
Lab Work!

- Create a procedure named **AgeGroup** that **takes a patient ID**, and displays his/her age group according to the following conditions:
  - Age > 60 → **'Old'**
  - Age >= 40 **and** Age <= 60 → **'Middle-Aged'**
  - Age < 40 → **'Young'**

Patient

PID	PName	Age	Payment
1	Mike	26	300
2	Kate	64	250
3	John	29	1500

- For example, by taking ID=2, the procedure displays 'Old'.



11

Lab Work!

```
delimiter //
create procedure AgeGroup(IN ID int, OUT ageLevel varchar(10))
begin
    declare patientAge int;
    select age into patientAge from patient where PID=ID;
    select
        case
            when patientAge>60 then 'Old'
            when patientAge between 40 and 60 then 'Middle-Aged'
            else 'Young'
        end
    into ageLevel;
end//
delimiter ;
```

Step 1 – Creating Function

```
set @ageLevel = "";
call AgeGroup(2,@ageLevel);
select @ageLevel;
```

Step 2 – Calling Function

@ageLevel
Old

Patient

PID	PName	Age	Payment
1	Mike	26	300
2	Kate	64	250
3	John	29	1500

12