```sql
CREATE TABLE StuffType
(
  TypeID INT NOT NULL,
  TypeName VARCHAR(10) NOT NULL,
  CONSTRAINT pk_StuffType PRIMARY KEY (TypeID)
);
GO

CREATE TABLE OurStuff
(
  StuffID INT NOT NULL PRIMARY KEY,
  StuffName VARCHAR(10) NOT NULL,
  OurTypeID INT NULL
  CONSTRAINT fk_StuffType FOREIGN KEY (OurTypeID)
    REFERENCES StuffType(TypeID)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
GO
```

Stored routine
SQL statements resueable and callable logic - functions and procedures
Proecudre is a subprogram that performs an action and provides reusability, modularity and maintainability.
Both has a block of SQL statement.

Stored fun must return a scalar value
Proc doesn't have to
Functions - no result set(name, lane with multiple rows), transactional commits and call backs.
Stored functions are called with Select while procedure are called with call.
Func - simple numerical functions, string manipulations etc.

Ad.
recompiled executed - they are ta the server.
reduced client/server traffic
reuse the code
Enhanced security control - may include that

STORED PROCEDURE per database - performs an action

SQL statements
Variable definition
Conditional statements
Loops Handlers
BEGIN and END required.

use mysql;
show tables;
describe db;
show grants;
ALTER/CREATE Routine privileges are stored in 'mysql.user' (global permission) and 'mysql.db' t (per database permission) tables
CALL db_name.routine_name(); - routine is asociated with a database
Stored routine implicitly execute 'use db_name' this means that 'USE db_ name' is not permissible within defined routine;
used for commonly executed queries;
SHOW PROECDURE STATUS;
SELECT * from proc; displays all procedures

Questions: if multiple records are returned?
Example:

DROP database mybd;
create database mydb;

1.
file name is procedure_school.sql

CREATE PROCEDURE procedure_name (IN | OUT | INOUT varibale_name TYPE) - variable scalar
DELIMITER $$ - something other than convention (;)
BEGIN

```
CREATE TABLE school_table (
        school_id NOT NULL,
        school_name VARCHAR(45) NOT NULL,
        PRIMARY KEY (school_id)
        );
```

END $$

CREATE PROCEDURE drop_school_table()

BEGIN
DROP TABLE school_table;
END $$

DELIMITER;

run sql file is : SOURCE person_data.sql - going to run sql command in the .sql file creates person table with 5 records.

Source procedure_school.sql

```
DELIMITER $$
CREATE PROCEDURE count_people()
BEGIN
SELECT COUNT(person_id) FROM person;
END $$
DELIMITER ;
```

CALL count_people; or CALL count_people(); - result same

SHOW CREATE PROCEDURE count_people\G - shows the content of procedure.
SHOW PROCEDURE STATUS LIKE '%school% \G

show tables; **OR show tables ?(without ;) - it is fine**
CALL create_school_table();
CALL drop_school_table();

2. IN (default), OUT, INOUT

IN

```
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 ;
```

OUT

```
DELIMITER $$
```

```
CREATE PROCEDURE get_person_name(IN p_id SMALLINT, OUT f_name
VARCHAR(45))

BEGIN
        SELECT f
```**irst_name INTO f_name** FROM person
```
        WHERE person_id = p_id;
END   $$
DELIMITER ;

CALL get_person_name(3, @myname);
SELECT @myname; -display value
@ - used for defining a variable;
```**I****NOUT**

```
DELIMITER $$

CREATE PROCEDURE compute_squre(INOUT number INT)
BEGIN
SELECT number * number INTO number;
END $$

```**SET @var = 7;**
```
CALL compute_square(@var);
SELECT @var;
```
——
STORED FUNCTIONS (always IN parameter) - competes a value - needs atleast one IN) - quick way f gaining access to scalar value

```
CREATE FUNCTION function_name(IN function_parameters ) RETURNS type
RETURN function_operations

Built-in function
function_parameters - parameter_name and parameter type
```**select current_user(); logged in as**
select database(); logged in database

```
DELIMITER $$

CREATE FUNCTION compute_square(number INT)
```**RETURNS INT**
```
BEGIN
        RETURN number*number;
END $$

CREATE FUNCTION compute_circle(radius INT)
```

RETURNS FLOAT

```
BEGIN
        RETURN PI() * radius*radius;
END $$

DELIMITER;

SELECT compute_function(3);
SELECT compute_circle(3);
```

EXAMPLE:

```
DELIMITER $$
CREATE FUNCTION sf_hello_whatever (arg1 char(10)) RETURNS char(20)
BEGIN
RETURN CONCAT('hello',arg1);
END $$
DELIMITER ;

SHOW FUNCTIONS status;

SELECT sf_hello_whatever(' world');
```

Create functions that will return the employee's age in years and days

```
DELIMITER $$

CREATE FUNCTION function_name sf_ age_in_years(d DATE)
RETURNS INT
BEGIN

RETURN DATEDIFF(now(), d) / 366 ;

END $$
DELIMITER ;

SELECT * from employees;

SELECT DATEDIFF('20160307','1975-08-20'); returns number of days.
SELECT DATEDIFF('20160307','1975-08-20')/366 ; returns a number say 30.4372
SELECT DATEDIFF(now(),'1975-08-20')/366 ; returns a number say 30.4372


show function status;
SELECT sf_age_in_years('1975/08/20');
```

```
DROP FUNCTION sf_age_in_years;
CREATE FUNCTION function_name sf_ age_in_years(d DATE, days_in_year INT)
RETURNS INT
BEGIN

RETURN DATEDIFF(now(), d) / days_in_year ;

END $$
DELIMITER ;

SELECT sf_age_in_years('1975/08/20','365');


DROP FUNCTION sf_age_in_years;
CREATE FUNCTION function_name sf_ age_in_years(d DATE, days_in_year INT)
RETURNS DECIMAL(6,3) (6 total numbers, 3 precision)
BEGIN

RETURN DATEDIFF(now(), d) / days_in_year ;

END $$
DELIMITER ;
```

————————————————————————

CASE Statement:

```
SELECT *,

CASE counter WHEN 1 THEN 'one' WHEN 5 THEN 'five ELSE 'Too high' END AS
result

FROM table1;


SELECT
        product.id,
        product_price.currency AS currency,
        product_price.price AS price,
        product_price.currency AS foreigncurrency,
        product_price.price AS foreginprice

        IF (forgeinPrice.currency IS NULL,
product_price.product_id,foreginPrice.curerncy) AS currency
FROM product

LEFT JOIN product_price ON (product.id = product_price.id)
```

```
LEFT JOIN product_price AS foreignPrice ON (product.id = product_price.id &&
foreignPrice.currency = ‘EUR)

WHERE product.id  = ‘1’ ’;

GROUP BY product.id
```