17. **Write a Function to generate Fibonacci series**

```
USE exam;

DELIMITER //

CREATE FUNCTION fibonacci(n INT) RETURNS INT DETERMINISTIC
BEGIN
   DECLARE a INT DEFAULT 0;
   DECLARE b INT DEFAULT 1;
   DECLARE i INT DEFAULT 0;
   DECLARE temp INT;

   IF n = 0 THEN
      RETURN 0;
   ELSEIF n = 1 THEN
      RETURN 1;
   ELSE
      WHILE i < n - 1 DO
         SET temp = b;
         SET b = a + b;
         SET a = temp;
         SET i = i + 1;
      END WHILE;
      RETURN b;
   END IF;
END //

DELIMITER ;

SELECT fibonacci(2);
```

Write a function to generate n factorial number

```
USE exam;

DELIMITER //

CREATE FUNCTION calculate_factorial(n INT) RETURNS BIGINT DETERMINISTIC
BEGIN
   DECLARE result BIGINT DEFAULT 1;
   DECLARE i INT DEFAULT 1;

   WHILE i <= n DO
      SET result = result * i;
      SET i = i + 1;
   END WHILE;
   RETURN result;
END //
DELIMITER ;

select calculate_factorial(5);
```

18 Write a function to generate sum of first n number

```
use exam;
DELIMITER //

CREATE FUNCTION SumOfFirstN(n INT) RETURNS INT DETERMINISTIC
BEGIN
   DECLARE sum_result INT DEFAULT 0;
   DECLARE counter INT DEFAULT 1;

   WHILE counter <= n DO
      SET sum_result = sum_result + counter;
      SET counter = counter + 1;
   END WHILE;

   RETURN sum_result;
END //

DELIMITER ;

SELECT SumOfFirstN(10) AS SumResult;
```

Write a function to check the given no. is prime or not

```
use exam;
DELIMITER //

CREATE FUNCTION IsPrime(n INT) RETURNS BOOLEAN DETERMINISTIC
BEGIN
   DECLARE i INT DEFAULT 2;

   IF n < 2 THEN
      RETURN FALSE; -- Numbers less than 2 are not prime
   END IF;

   WHILE i <= SQRT(n) DO
      IF n % i = 0 THEN
         RETURN FALSE; -- If n is divisible by i, it's not prime
      END IF;
      SET i = i + 1;
   END WHILE;

   RETURN TRUE; -- If no divisors found, the number is prime
END //

DELIMITER ;

SELECT IsPrime(4) AS IsPrimeResult;
```

19 **Write a function which print the sum of all even number between 1 to 100**

```
use exam;
DELIMITER //

CREATE FUNCTION SumOfEvenNumbers() RETURNS INT DETERMINISTIC
BEGIN
   DECLARE sum_result INT DEFAULT 0;
   DECLARE counter INT DEFAULT 2;

   WHILE counter <= 100 DO
      SET sum_result = sum_result + counter;
      SET counter = counter + 2;
   END WHILE;

   RETURN sum_result;
END //

DELIMITER ;

select SumOfEvenNumbers() AS SumOfEven;
```

20 **Write a function which accept i/o as number & print Whether it is Even or Odd**

```
use exam;
DELIMITER //

CREATE FUNCTION EvenOrOdd(input_number INT) RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
   DECLARE result VARCHAR(20);

   IF input_number % 2 = 0 THEN
      SET result = 'Even';
   ELSE
      SET result = 'Odd';
   END IF;

   RETURN result;
END //

DELIMITER ;

SELECT  EvenOrOdd(10) AS Result;
```

21 Write function to calculate income tax , pass basic (per month) as input DA = 12% of Basic, HRA = 10% of Basic, TA = 15 % of Basic, PF = 8% of Basic. Income tax on Annual Income slabs are as follows upto 1 Lack – Nil , 100000 to 150000 – 10%, 150000 to 250000 – 15%, < 250000 – 20%

```
use exam;
DELIMITER //

CREATE FUNCTION CalculateIncomeTax(basic_salary DECIMAL(10, 2))
RETURNS DECIMAL(10, 2) DETERMINISTIC
BEGIN
   DECLARE da DECIMAL(10, 2);
   DECLARE hra DECIMAL(10, 2);
   DECLARE ta DECIMAL(10, 2);
   DECLARE pf DECIMAL(10, 2);
   DECLARE annual_income DECIMAL(10, 2);
   DECLARE income_tax DECIMAL(10, 2);

   -- Calculate allowances
   SET da = 0.12 * basic_salary;
   SET hra = 0.10 * basic_salary;
   SET ta = 0.15 * basic_salary;
   SET pf = 0.08 * basic_salary;

   -- Calculate annual income
   SET annual_income = 12 * (basic_salary + da + hra + ta - pf);

   -- Calculate income tax based on slabs
   IF annual_income <= 100000 THEN
      SET income_tax = 0;
   ELSEIF annual_income <= 150000 THEN
      SET income_tax = 0.10 * (annual_income - 100000);
   ELSEIF annual_income <= 250000 THEN
      SET income_tax = 0.10 * 50000 + 0.15 * (annual_income - 150000);
   ELSE
      SET income_tax = 0.10 * 50000 + 0.15 * 100000 + 0.20 * (annual_income -
250000);
   END IF;

   RETURN income_tax;
END //

DELIMITER ;

SELECT CalculateIncomeTax(50000) AS IncomeTax;
```

22. **For an employee database raise the salary by 5 %**
For all Manager assume { emp(emp_no , name , designation , salary)

```
UPDATE employee
SET salary = salary * 1.05
WHERE designation = 'Manager';
```

23**. Procedure for reversing the given number or string**

For Number:

```
use exam;
DELIMITER //

CREATE PROCEDURE ReverseNumber(INOUT input_num INT)
BEGIN
   DECLARE reversed_num INT DEFAULT 0;
   DECLARE remainder INT;

   WHILE input_num > 0 DO
      SET remainder = input_num % 10;
      SET reversed_num = reversed_num * 10 + remainder;
      SET input_num = input_num DIV 10;
   END WHILE;

   SET input_num = reversed_num;
END //

DELIMITER ;

SET @number_to_reverse = 12345;
CALL ReverseNumber(@number_to_reverse);
SELECT @number_to_reverse AS ReversedNumber;
```

For string:

```
use exam;
DELIMITER //

CREATE PROCEDURE ReverseString(INOUT input_str VARCHAR(255))
BEGIN
   DECLARE reversed_str VARCHAR(255) DEFAULT '';
   DECLARE str_length INT;

   SET str_length = LENGTH(input_str);

   WHILE str_length > 0 DO
      SET reversed_str = CONCAT(reversed_str, SUBSTRING(input_str, str_length, 1));
      SET str_length = str_length - 1;
   END WHILE;
```

```
    SET input_str = reversed_str;
END //

DELIMITER ;

SET @string_to_reverse = 'Hello';
CALL ReverseString(@string_to_reverse);
SELECT @string_to_reverse AS ReversedString;
```

24. **Write a trigger which will performs**
    If insert then display total number of rows in database before insert
    If updating then should not allow sal > 9000
    If deleting then display message that row is deleted.

```
DELIMITER //

CREATE TRIGGER Before_Insert_Update_Delete
BEFORE INSERT OR UPDATE OR DELETE ON your_table
FOR EACH ROW
BEGIN
    DECLARE total_rows INT;

    IF (INSERTING) THEN
        -- Display total number of rows before insert
        SELECT COUNT(*) INTO total_rows FROM your_table;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = CONCAT('Total rows
before insert: ', total_rows);
    END IF;

    IF (UPDATING) THEN
        -- Check if the updated salary is not greater than 9000
        IF NEW.sal > 9000 THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Salary cannot be
greater than 9000 during update.';
        END IF;
    END IF;

    IF (DELETING) THEN
        -- Display a message that a row is deleted
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Row is deleted.';
    END IF;
END //

DELIMITER ;
```

25. **Write trigger for Audit trail.**

```
create table company(name varchar(12),dept_no number(5));
create table company_log(who varchar(12),when date,action varchar(9));
```

```
DELIMITER //

CREATE TRIGGER Company_Audit_Trigger
AFTER INSERT OR UPDATE OR DELETE ON company
FOR EACH ROW
BEGIN
   DECLARE action_description VARCHAR(9);

   IF (INSERTING) THEN
      SET action_description = 'INSERT';
   ELSEIF (UPDATING) THEN
      SET action_description = 'UPDATE';
   ELSE
      SET action_description = 'DELETE';
   END IF;

   INSERT INTO company_log (who, when, action)
   VALUES (USER(), NOW(), action_description);
END //

DELIMITER ;
```

Function to square the number taken from user

```
use exam;
delimiter //
create function SquareNumber(input_number decimal(10,2))
returns decimal(10,2)
deterministic
begin
declare squared decimal(10,2);
set squared = input_number * input_number;
return squared;
end;
//

select SquareNumber(5);
```