**MySQL Exercise 3**

1.

DELIMITER //

CREATE PROCEDURE iterate\_and\_print()

BEGIN

DECLARE i INT DEFAULT 1;

WHILE i <= 1000 DO

IF i MOD 100 = 0 THEN

SELECT i;

END IF;

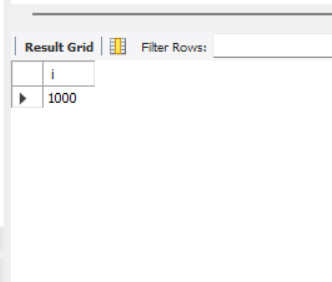
SET i = i + 1;

END WHILE;

END //

DELIMITER ;

CALL iterate\_and\_print();



2.

DELIMITER //

CREATE PROCEDURE find\_special\_numbers()

BEGIN

DECLARE i INT DEFAULT 1;

DECLARE num INT;

DECLARE digit1 INT;

DECLARE digit2 INT;

DECLARE digit3 INT;

DECLARE sum\_of\_cubes INT;

DROP TABLE IF EXISTS SpecialNumbers;

CREATE TABLE SpecialNumbers (

Number INT

);

WHILE i <= 999 DO

SET num = i;

SET digit1 = num DIV 100; -- Hundreds place

SET digit2 = (num DIV 10) MOD 10; -- Tens place

SET digit3 = num MOD 10; -- Ones place

SET sum\_of\_cubes = POW(digit1, 3) + POW(digit2, 3) + POW(digit3, 3);

IF sum\_of\_cubes = i THEN

INSERT INTO SpecialNumbers (Number) VALUES (i);

END IF;

SET i = i + 1;

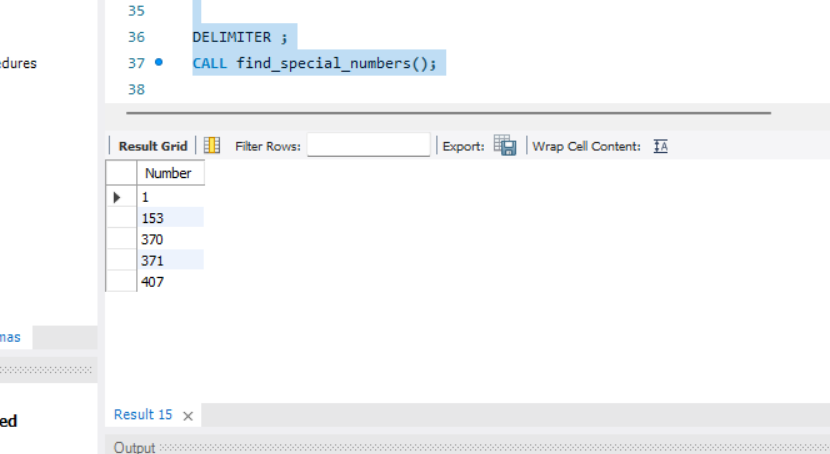
END WHILE;

SELECT \* FROM SpecialNumbers;

END //

DELIMITER ;

CALL find\_special\_numbers();



3.

CREATE TABLE RadiusConfig (

id INT AUTO\_INCREMENT PRIMARY KEY,

min\_radius FLOAT NOT NULL,

max\_radius FLOAT NOT NULL,

increment FLOAT NOT NULL

);

-- Insert a sample configuration

INSERT INTO RadiusConfig (min\_radius, max\_radius, increment) VALUES (1.0, 5.0, 0.5);

DELIMITER //

CREATE PROCEDURE generate\_sphere\_metrics()

BEGIN

DECLARE r FLOAT;

DECLARE min\_r FLOAT;

DECLARE max\_r FLOAT;

DECLARE inc FLOAT;

-- Select the min, max, and increment from the configuration table

SELECT min\_radius, max\_radius, increment INTO min\_r, max\_r, inc FROM RadiusConfig LIMIT 1;

-- Drop the temporary results table if it exists

DROP TABLE IF EXISTS SphereMetrics;

-- Create a temporary table to store the results

CREATE TABLE SphereMetrics (

radius FLOAT,

circumference FLOAT,

area FLOAT,

volume FLOAT

);

-- Initialize the radius with the minimum radius

SET r = min\_r;

-- Loop to calculate the metrics for each radius

WHILE r <= max\_r DO

INSERT INTO SphereMetrics (radius, circumference, area, volume)

VALUES (

r,

2 \* PI() \* r, -- Circumference

4 \* PI() \* POW(r, 2), -- Surface Area

(4 / 3) \* PI() \* POW(r, 3) -- Volume

);

SET r = r + inc;

END WHILE;

-- Include the maximum value

IF r - inc < max\_r THEN

INSERT INTO SphereMetrics (radius, circumference, area, volume)

VALUES (

max\_r,

2 \* PI() \* max\_r, -- Circumference

4 \* PI() \* POW(max\_r, 2), -- Surface Area

(4 / 3) \* PI() \* POW(max\_r, 3) -- Volume

);

END IF;

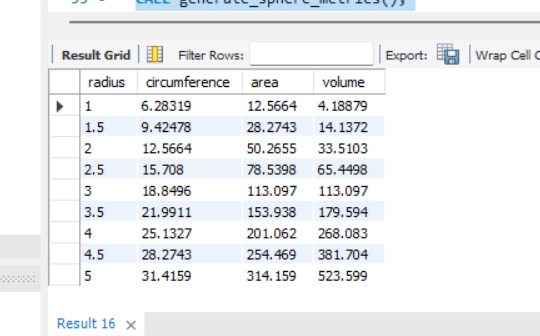
-- Select the results from the temporary table

SELECT \* FROM SphereMetrics;

END //

DELIMITER ;

CALL generate\_sphere\_metrics();



4.

CREATE TABLE Words (

id INT AUTO\_INCREMENT PRIMARY KEY,

word VARCHAR(5) NOT NULL

);

-- Insert some sample words

INSERT INTO Words (word) VALUES ('level'), ('radar'), ('hello'), ('world'), ('refer');

DELIMITER //

CREATE PROCEDURE check\_palindromes()

BEGIN

DECLARE word VARCHAR(5);

DECLARE reversed\_word VARCHAR(5);

DECLARE finished INTEGER DEFAULT 0;

-- Declare a cursor to select five-letter words

DECLARE word\_cursor CURSOR FOR

SELECT word FROM Words WHERE CHAR\_LENGTH(word) = 5;

-- Declare a NOT FOUND handler for the cursor

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

-- Drop the temporary results table if it exists

DROP TABLE IF EXISTS Palindromes;

-- Create a temporary table to store the results

CREATE TABLE Palindromes (

word VARCHAR(5),

is\_palindrome BOOLEAN

);

-- Open the cursor

OPEN word\_cursor;

read\_loop: LOOP

FETCH word\_cursor INTO word;

-- Exit the loop if no more rows

IF finished THEN

LEAVE read\_loop;

END IF;

-- Reverse the word

SET reversed\_word = REVERSE(word);

-- Check if the word is a palindrome

IF word = reversed\_word THEN

INSERT INTO Palindromes (word, is\_palindrome) VALUES (word, TRUE);

ELSE

INSERT INTO Palindromes (word, is\_palindrome) VALUES (word, FALSE);

END IF;

END LOOP;

-- Close the cursor

CLOSE word\_cursor;

-- Select the results from the Palindromes table

SELECT \* FROM Palindromes;

END //

DELIMITER ;

CALL check\_palindromes();

5.

CREATE TABLE Words (

id INT AUTO\_INCREMENT PRIMARY KEY,

word VARCHAR(255) NOT NULL

);

-- Insert some sample words

INSERT INTO Words (word) VALUES ('level'), ('radar'), ('hello'), ('world'), ('refer'), ('noon'), ('palindrome');

DELIMITER //

CREATE PROCEDURE check\_variable\_length\_palindromes()

BEGIN

DECLARE word VARCHAR(255);

DECLARE reversed\_word VARCHAR(255);

DECLARE finished INTEGER DEFAULT 0;

-- Declare a cursor to select words from the Words table

DECLARE word\_cursor CURSOR FOR

SELECT word FROM Words;

-- Declare a NOT FOUND handler for the cursor

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

-- Drop the temporary results table if it exists

DROP TABLE IF EXISTS Palindromes;

-- Create a temporary table to store the results

CREATE TABLE Palindromes (

word VARCHAR(255),

is\_palindrome BOOLEAN

);

-- Open the cursor

OPEN word\_cursor;

read\_loop: LOOP

FETCH word\_cursor INTO word;

-- Exit the loop if no more rows

IF finished THEN

LEAVE read\_loop;

END IF;

-- Reverse the word

SET reversed\_word = REVERSE(word);

-- Check if the word is a palindrome

IF word = reversed\_word THEN

INSERT INTO Palindromes (word, is\_palindrome) VALUES (word, TRUE);

ELSE

INSERT INTO Palindromes (word, is\_palindrome) VALUES (word, FALSE);

END IF;

END LOOP;

-- Close the cursor

CLOSE word\_cursor;

-- Select the results from the Palindromes table

SELECT \* FROM Palindromes;

END //

DELIMITER ;

CALL check\_variable\_length\_palindromes();

