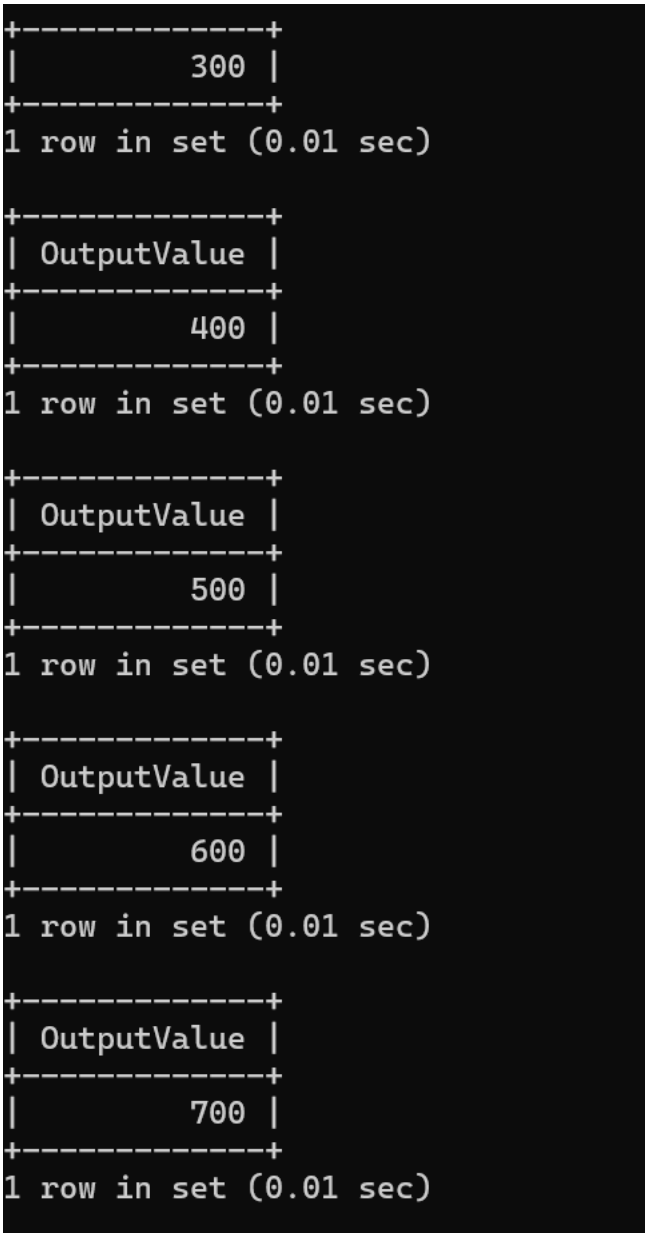
MySQL Exercise 3

1. Write a program containing a loop that iterates from 1 to 1000 using a variable I, which is incremented each time around the loop. The program should output the value of I every hundred iterations (i.e., the output should be 100, 200, etc.).

```
D2_92814_Krushna>DELIMITER //
D2_92814_Krushna>
D2_92814_Krushna>CREATE PROCEDURE LoopHundreds()
    -> BEGIN
    ->     DECLARE i INT DEFAULT 1;
    ->
    ->     WHILE i <= 1000 DO
    ->         IF i % 100 = 0 THEN
    ->             SELECT i AS OutputValue;
    ->         END IF;
    ->         SET i = i + 1;
    ->     END WHILE;
    -> END//
Query OK, 0 rows affected (0.04 sec)

D2_92814_Krushna>
D2_92814_Krushna>DELIMITER ;
D2_92814_Krushna>CALL LoopHundreds();
+-------------+
| OutputValue |
+-------------+
|         100 |
+-------------+
1 row in set (0.00 sec)

+-------------+
| OutputValue |
+-------------+
|         200 |
+-------------+
1 row in set (0.01 sec)
```



2. Write a program that examines all the numbers from 1 to 999, displaying all those for which the sum of the cubes of the digits equal the number itself.

```
D2_92814_Krushna>CREATE PROCEDURE FindArmstrongNumbers()
    -> BEGIN
    ->     DECLARE i INT DEFAULT 1;
    ->     DECLARE d1 INT;
    ->     DECLARE d2 INT;
    ->     DECLARE d3 INT;
    ->     DECLARE sum_cubes INT;
    ->
    ->     WHILE i <= 999 DO
    ->         -- Extract digits
    ->         SET d1 = i % 10;
    ->         SET d2 = (i / 10) % 10;
    ->         SET d3 = (i / 100) % 10;
    ->
    ->         -- Calculate sum of cubes
    ->         SET sum_cubes = POW(d1,3) + POW(d2,3) + POW(d3,3);
    ->
    ->         -- Check if equal to the number
    ->         IF sum_cubes = i THEN
    ->             SELECT i AS ArmstrongNumber;
    ->         END IF;
    ->
    ->         SET i = i + 1;
    ->     END WHILE;
    -> END//
Query OK, 0 rows affected (0.05 sec)
```

```
D2_92814_Krushna>CALL FindArmstrongNumbers();
+-----------------+
| ArmstrongNumber |
+-----------------+
|               1 |
+-----------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.02 sec)
```

3. Write a program that Selects from any table a minimum and maximum value for a radius, along with an increment factor, and generates a series of radii by repeatedly adding the increment to the minimum until the maximum is reached. For each value of the radius, compute and display the circumference, area, and volume of the sphere. (Be sure to include both the maximum and the minimum values.).

```
D2_92814_Krushna>CREATE PROCEDURE calc_sphere()
    -> BEGIN
    ->     DECLARE v_min   FLOAT;
    ->     DECLARE v_max   FLOAT;
    ->     DECLARE v_inc   FLOAT;
    ->     DECLARE v_r     FLOAT;
    ->     DECLARE v_circ  FLOAT;
    ->     DECLARE v_area  FLOAT;
    ->     DECLARE v_vol   FLOAT;
    ->
    ->     SELECT min_radius, max_radius, increment
    ->     INTO v_min, v_max, v_inc
    ->     FROM radius_values
    ->     LIMIT 1;
    ->
    ->     SET v_r = v_min;
    ->     WHILE v_r <= v_max DO
    ->         SET v_circ = 2 * PI() * v_r;
    ->         SET v_area = 4 * PI() * POW(v_r, 2);
    ->         SET v_vol  = (4/3) * PI() * POW(v_r, 3);
    ->
    ->         SELECT CONCAT('Radius = ', v_r,
    ->                       ' | Circumference = ', ROUND(v_circ,2),
    ->                       ' | Area = ', ROUND(v_area,2),
    ->                       ' | Volume = ', ROUND(v_vol,2));
    ->
    ->         SET v_r = v_r + v_inc;
    ->     END WHILE;
    -> END //
Query OK, 0 rows affected (0.05 sec)
```

```
D2_92814_Krushna>CALL calc_sphere();
+-------------------------------------------------------------------------
-------------------------------------------------------------+
| CONCAT('Radius = ', v_r,
                     ' | Circumference = ', ROUND(v_circ,2),
                     ' | Area = ', ROUND(v_area,2),
                     ' | Volume = ', ROUND(v_vol,2)) |
+-------------------------------------------------------------------------
-------------------------------------------------------------+
| Radius = 1 | Circumference = 6.28 | Area = 12.57 | Volume = 4.19
                                                                |
+-------------------------------------------------------------------------
-------------------------------------------------------------+
1 row in set (0.00 sec)

+-------------------------------------------------------------------------
-------------------------------------------------------------+
| CONCAT('Radius = ', v_r,
                     ' | Circumference = ', ROUND(v_circ,2),
                     ' | Area = ', ROUND(v_area,2),
                     ' | Volume = ', ROUND(v_vol,2)) |
+-------------------------------------------------------------------------
-------------------------------------------------------------+
| Radius = 2 | Circumference = 12.57 | Area = 50.27 | Volume = 33.51
                                                                |
+-------------------------------------------------------------------------
-------------------------------------------------------------+
1 row in set (0.01 sec)

+-------------------------------------------------------------------------
```

4. A palindrome is a word that is spelled the same forward and backward, such as level, radar, etc. Write a program to Selects from any table a five letter word and determine whether it is a palindrome.

```
D2_92814_Krushna>CREATE PROCEDURE check_palindrome()
    -> BEGIN
    ->     DECLARE done INT DEFAULT 0;
    ->     DECLARE v_word VARCHAR(20);
    ->     DECLARE v_rev  VARCHAR(20);
    ->
    ->     DECLARE cur CURSOR FOR SELECT word FROM words;
    ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    ->
    ->     OPEN cur;
    ->     read_loop: LOOP
    ->         FETCH cur INTO v_word;
    ->         IF done = 1 THEN
    ->             LEAVE read_loop;
    ->         END IF;
    ->
    ->         SET v_rev = REVERSE(v_word);
    ->
    ->         IF v_word = v_rev THEN
    ->             SELECT CONCAT(v_word, ' is a palindrome') AS result;
    ->         ELSE
    ->             SELECT CONCAT(v_word, ' is not a palindrome') AS result;
    ->         END IF;
    ->     END LOOP;
    ->
    ->     CLOSE cur;
    -> END //
Query OK, 0 rows affected (0.05 sec)
```

```
D2_92814_Krushna>CALL check_palindrome();
+----------------------+
| result               |
+----------------------+
| LEVEL is a palindrome |
+----------------------+
1 row in set (0.04 sec)

+----------------------+
| result               |
+----------------------+
| RADAR is a palindrome |
+----------------------+
1 row in set (0.05 sec)

+--------------------------+
| result                   |
+--------------------------+
| HELLO is not a palindrome |
+--------------------------+
1 row in set (0.05 sec)

Query OK, 0 rows affected (0.06 sec)
```

5. Modify the above program to Select from any table a variable length word. This requires determining how many characters are read in.

```
D2_92814_Krushna>CREATE PROCEDURE check_palindrome_var()
    -> BEGIN
    ->     DECLARE done INT DEFAULT 0;
    ->     DECLARE v_word VARCHAR(100);
    ->     DECLARE v_rev  VARCHAR(100);
    ->
    ->     DECLARE cur CURSOR FOR SELECT word FROM words;
    ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    ->
    ->     OPEN cur;
    ->     read_loop: LOOP
    ->         FETCH cur INTO v_word;
    ->         IF done = 1 THEN
    ->             LEAVE read_loop;
    ->         END IF;
    ->
    ->         SET v_rev = REVERSE(v_word);
    ->
    ->         IF v_word = v_rev THEN
    ->             SELECT CONCAT(v_word, ' is a palindrome') AS result;
    ->         ELSE
    ->             SELECT CONCAT(v_word, ' is not a palindrome') AS result;
    ->         END IF;
    ->     END LOOP;
    ->
    ->     CLOSE cur;
    -> END //
Query OK, 0 rows affected (0.04 sec)
```

```
D2_92814_Krushna>CALL check_palindrome_var();
+----------------------+
| result               |
+----------------------+
| LEVEL is a palindrome |
+----------------------+
1 row in set (0.00 sec)

+----------------------+
| result               |
+----------------------+
| RADAR is a palindrome |
+----------------------+
1 row in set (0.01 sec)

+--------------------------+
| result                   |
+--------------------------+
| HELLO is not a palindrome |
+--------------------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```