PL 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.).

```
mysql> CREATE TABLE tempp (
   ->     value INT
   -> );
Query OK, 0 rows affected (0.04 sec)

mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE fill_tempp()
   -> BEGIN
   ->     DECLARE i INT DEFAULT 1;
   ->
   ->     WHILE i <= 1000 DO
   ->         IF MOD(i, 100) = 0 THEN
   ->             INSERT INTO tempp (value) VALUES (i);
   ->         END IF;
   ->         SET i = i + 1;
   ->     END WHILE;
   -> END;
   -> //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL fill_tempp();
Query OK, 1 row affected (0.03 sec)

mysql> SELECT * FROM tempp;
+-------+
| value |
+-------+
|   100 |
|   200 |
|   300 |
|   400 |
|   500 |
|   600 |
|   700 |
|   800 |
|   900 |
|  1000 |
+-------+
10 rows in set (0.00 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.

```
mysql> CREATE TABLE tempp (
    ->     number INT
    -> );
Query OK, 0 rows affected (0.07 sec)

mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE find_armstrong_numbers()
    -> BEGIN
    ->     DECLARE i INT DEFAULT 1;
    ->     DECLARE d1 INT;
    ->     DECLARE d2 INT;
    ->     DECLARE d3 INT;
    ->     DECLARE sum_of_cubes INT;
    ->
    ->     WHILE i <= 999 DO
    ->         SET d1 = FLOOR(i / 100);            -- Hundreds digit
    ->         SET d2 = FLOOR((i % 100) / 10);       -- Tens digit
    ->         SET d3 = i % 10;                    -- Units digit
    ->
    ->         SET sum_of_cubes = POW(d1, 3) + POW(d2, 3) + POW(d3, 3);
    ->
    ->         IF sum_of_cubes = i THEN
    ->             INSERT INTO tempp (number) VALUES (i);
    ->         END IF;
    ->
    ->         SET i = i + 1;
    ->     END WHILE;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql>
mysql> CALL find_armstrong_numbers();
Query OK, 1 row affected (0.04 sec)

mysql> SELECT * FROM tempp;
+--------+
| number |
+--------+
|      1 |
|    153 |
|    370 |
|    371 |
|    407 |
+--------+
5 rows in set (0.00 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.).

```
mysql> CREATE TABLE radius_settings (
    ->     min_radius FLOAT,
    ->     max_radius FLOAT,
    ->     increment FLOAT
    -> );
Query OK, 0 rows affected (0.07 sec)

mysql> INSERT INTO radius_settings VALUES (1, 5, 1);
Query OK, 1 row affected (0.00 sec)

mysql> CREATE TABLE sphere_results (
    ->     radius FLOAT,
    ->     circumference FLOAT,
    ->     area FLOAT,
    ->     volume FLOAT
    -> );
Query OK, 0 rows affected (0.12 sec)

mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE generate_sphere_calculations()
    -> BEGIN
    ->     DECLARE r FLOAT;
    ->     DECLARE min_r FLOAT;
    ->     DECLARE max_r FLOAT;
    ->     DECLARE inc FLOAT;
    ->     DECLARE pi FLOAT DEFAULT 3.14159;
    ->
    ->     -- Get settings from the table
    ->     SELECT min_radius, max_radius, increment
    ->     INTO min_r, max_r, inc
    ->     FROM radius_settings
    ->     LIMIT 1;
    ->
    ->     SET r = min_r;
    ->
    ->     WHILE r <= max_r DO
    ->         INSERT INTO sphere_results (radius, circumference, area, volume)
    ->         VALUES (
    ->             r,
    ->             2 * pi * r,
    ->             pi * POW(r, 2),
    ->             (4.0 / 3.0) * pi * POW(r, 3)
    ->         );
    ->         SET r = r + inc;
    ->     END WHILE;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql>
```

```
mysql> DELIMITER ;
mysql>
mysql> CALL generate_sphere_calculations();
Query OK, 1 row affected (0.09 sec)

mysql> SELECT * FROM sphere_results;
+--------+---------------+---------+---------+
| radius | circumference | area    | volume  |
+--------+---------------+---------+---------+
|      1 |       6.28318 | 3.14159 | 4.18879 |
|      2 |       12.5664 | 12.5664 | 33.5103 |
|      3 |       18.8495 | 28.2743 | 113.097 |
|      4 |       25.1327 | 50.2654 | 268.082 |
|      5 |       31.4159 | 78.5397 | 523.598 |
+--------+---------------+---------+---------+
5 rows in set (0.00 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.

```
mysql> CREATE TABLE word_table (
   ->     word VARCHAR(5)
   -> );
Query OK, 0 rows affected (0.14 sec)

mysql> INSERT INTO word_table VALUES ('level'), ('radar'), ('hello'), ('apple');
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> CREATE TABLE palindrome_check (
   ->     word VARCHAR(5),
   ->     is_palindrome VARCHAR(10)
   -> );
Query OK, 0 rows affected (0.13 sec)

mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE check_palindrome()
   -> BEGIN
   ->     DECLARE done INT DEFAULT 0;
   ->     DECLARE w VARCHAR(5);
   ->     DECLARE rev VARCHAR(5);
   ->     DECLARE cur CURSOR FOR SELECT word FROM word_table;
   ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
   ->
   ->     OPEN cur;
   ->
   ->     read_loop: LOOP
   ->         FETCH cur INTO w;
   ->         IF done THEN
   ->             LEAVE read_loop;
   ->         END IF;
   ->
   ->         SET rev = REVERSE(w);
   ->
   ->         IF w = rev THEN
   ->             INSERT INTO palindrome_check VALUES (w, 'Yes');
```

```
    ->         ELSE
    ->             INSERT INTO palindrome_check VALUES (w, 'No');
    ->         END IF;
    ->     END LOOP;
    ->
    ->     CLOSE cur;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL check_palindrome();
Query OK, 0 rows affected (0.02 sec)

mysql> SELECT * FROM palindrome_check;
+-------+---------------+
| word  | is_palindrome |
+-------+---------------+
| level | Yes           |
| radar | Yes           |
| hello | No            |
| apple | No            |
+-------+---------------+
4 rows in set (0.00 sec)
```

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

```
mysql> CREATE TABLE palindrome_check (
    ->     word VARCHAR(255),
    ->     is_palindrome VARCHAR(10),
    ->     length INT
    -> );
Query OK, 0 rows affected (0.12 sec)

mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE check_palindrome_variable_length()
    -> BEGIN
    ->     DECLARE done INT DEFAULT 0;
    ->     DECLARE w VARCHAR(255);
    ->     DECLARE rev VARCHAR(255);
    ->     DECLARE word_length INT;
    ->     DECLARE cur CURSOR FOR SELECT word FROM word_table;
    ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    ->
    ->     OPEN cur;
    ->
    ->     read_loop: LOOP
    ->         FETCH cur INTO w;
    ->         IF done THEN
    ->             LEAVE read_loop;
    ->         END IF;
    ->
    ->         -- Get the length of the word
    ->         SET word_length = LENGTH(w);
```

```
    ->
    ->     -- Reverse the word
    ->     SET rev = REVERSE(w);
    ->
    ->     -- Check if the word is a palindrome
    ->     IF w = rev THEN
    ->         INSERT INTO palindrome_check VALUES (w, 'Yes', word_length);
    ->     ELSE
    ->         INSERT INTO palindrome_check VALUES (w, 'No', word_length);
    ->     END IF;
    ->   END LOOP;
    ->
    ->   CLOSE cur;
    -> END;
    -> //
Query OK, 0 rows affected (0.11 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL check_palindrome_variable_length();
Query OK, 0 rows affected (0.11 sec)

mysql> SELECT * FROM palindrome_check;
+-------+---------------+--------+
| word  | is_palindrome | length |
+-------+---------------+--------+
| level | Yes           |      5 |
| radar | Yes           |      5 |
| hello | No            |      5 |
| apple | No            |      5 |
+-------+---------------+--------+
4 rows in set (0.00 sec)
```