Experiment – 12

Objective: Students will be able to implement the concept of sequence.

1. Create a sequence by name EMPID_SEQ starting with value 100 with an interval of 1.

```
CREATE SEQUENCE EMPID_SEQ

START WITH 100

INCREMENT BY 1;
```

2. Write a SQL command for finding the current and the next status of EMPID_SEQ.

```
SELECT EMPID_SEQ.NEXTVAL AS Next_Value FROM dual;

SELECT EMPID_SEQ.CURRVAL AS Current_Value FROM dual;
```

3. Change the Cache value of the sequence EMPID_SEQ to 20 and maxvalue to 1000.

```
ALTER SEQUENCE EMPID_SEQ

CACHE 20

MAXVALUE 1000;
```

4. Insert values in employees table using sequences for employee_id column.

```
SELECT EMPID_SEQ.NEXTVAL FROM dual;
```

5. Drop sequence EMPID_SEQ.

```
DROP SEQUENCE EMPID_SEQ;
```

6. Create a sequence called REVERSE to generate numbers in the descending order from 10000 to 1000 with a decrement of 5.

```
CREATE SEQUENCE REVERSE

START WITH 10000

INCREMENT BY -5

MINVALUE 1000;
```

Experiment – 13

```
create database exp13;

use exp13;
```

1. Write a PL/SQL code to accept the value of A, B & C display which is greater.

```
DELIMITER $$


CREATE PROCEDURE compare_values(A INT, B INT, C INT)

BEGIN

  IF A > B AND A > C THEN

    SELECT 'A is the greatest';

  ELSEIF B > A AND B > C THEN

    SELECT 'B is the greatest';

  ELSE

    SELECT 'C is the greatest';

  END IF;

END $$


DELIMITER ;
```

```
mysql> CALL compare_values(10, 20, 30);
+-------------------+
| C is the greatest |
+-------------------+
| C is the greatest |
+-------------------+
1 row in set (0.01 sec)

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

2. Using PL/SQL Statements create a simple loop that display message "Welcome to PL/SQL Programming" 20 times.

```
DELIMITER $$


CREATE PROCEDURE display_welcome()
BEGIN
  DECLARE counter INT DEFAULT 1;
  WHILE counter <= 20 DO
    SELECT 'Welcome to PL/SQL Programming';
    SET counter = counter + 1;
  END WHILE;
END $$


DELIMITER ;
CALL display_welcome();
```

3. Write a PL/SQL code block to find the factorial of a number.

```
DELIMITER $$
```

```
CREATE PROCEDURE factorial(num INT)
BEGIN
  DECLARE fact INT DEFAULT 1;
  DECLARE i INT DEFAULT 1;

  WHILE i <= num DO
    SET fact = fact * i;
    SET i = i + 1;
  END WHILE;

  SELECT CONCAT('Factorial of ', num, ' is: ', fact) AS Result;
END $$

DELIMITER ;
CALL factorial(5);
```

```
+-------------------------+
| Result                  |
+-------------------------+
| Factorial of 5 is: 120 |
+-------------------------+
1 row in set (0.00 sec)
```

4. Write a PL/SQL program to generate Fibonacci series.

```
DELIMITER $$

CREATE PROCEDURE fibonacci_series(n INT)
BEGIN
```

```
  DECLARE a INT DEFAULT 0;
  DECLARE b INT DEFAULT 1;
  DECLARE c INT;
  DECLARE counter INT DEFAULT 1;

  -- Display first two Fibonacci numbers
  SELECT a;
  SELECT b;

  -- Generate the remaining Fibonacci numbers
  WHILE counter < n DO
    SET c = a + b;
    SELECT c;
    SET a = b;
    SET b = c;
    SET counter = counter + 1;
  END WHILE;
END $$

DELIMITER ;
CALL fibonacci_series(10);
```

5. Write a PL/SQL code to fund the sum of first N numbers

```
DELIMITER $$

CREATE PROCEDURE sum_of_numbers(n INT)
```

```
BEGIN
  DECLARE sum INT DEFAULT 0;
  DECLARE i INT DEFAULT 1;

  WHILE i <= n DO
    SET sum = sum + i;
    SET i = i + 1;
  END WHILE;

  SELECT CONCAT('Sum of first ', n, ' numbers is: ', sum) AS Result;
END $$

DELIMITER ;
CALL sum_of_numbers(10);
```

```
+-------------------------------+
| Result                        |
+-------------------------------+
| Sum of first 10 numbers is: 55 |
+-------------------------------+
1 row in set (0.00 sec)
```