**Name:** Prashant Suresh Shirgave
**Roll No**:3 **Batch:** T1
**Class**: TY(CSE-AIML)

**Experiment No. 9**

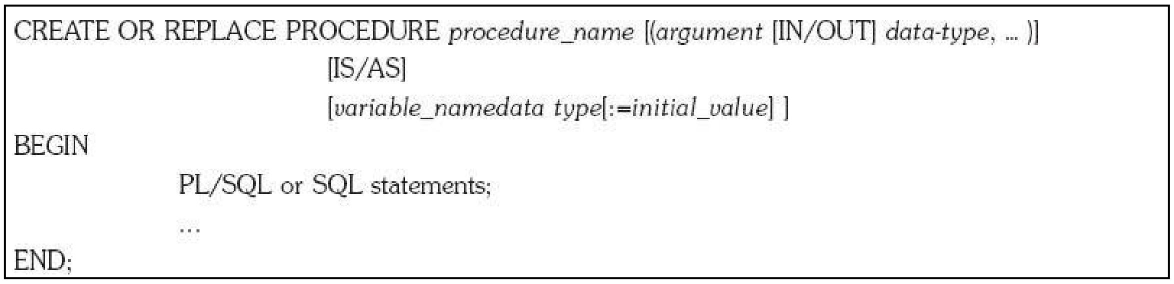
**Title**: PLSQL Functions and Procedures

**Objective**: Demonstrate PLSQL Functions and Procedures.

**Theory:**
**Stored Procedure:**
A stored procedure is a named collection of procedural and SQL statements.

There are two advantages to the use of stored procedures:
1. Stored procedures substantially reduce network traffic and increase performance. Because the procedure is stored at the server, there is no transmission of individual SQL statements over the network. The use of stored procedures improves system performance because all transactions are executed locally on the RDBMS, so each SQL statement does not have to travel over the network.
2. Stored procedures help reduce code duplication by means of code isolation and code sharing (creating unique PL/SQL modules that are called by application programs), thereby minimizing the chance of errors and the cost of application development and maintenance.

**Syntax to create a stored procedure(on Oracle database):**



```
CREATE OR REPLACE PROCEDURE procedure_name [(argument [IN/OUT] data-type, ... )]
                            [IS/AS]
                            [variable_namedata type[:=initial_value] ]
BEGIN
            PL/SQL or SQL statements;
            ...
END;
```

1. argument specifies the parameters that are passed to the stored procedure. A stored procedure could have zero or more arguments or parameters.

2. IN/OUT indicates whether the parameter is for input, output, or both.

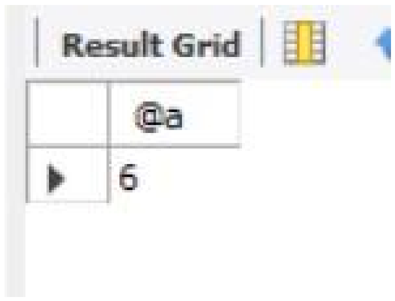
3.data-type is one of the procedural SQL data types used in the RDBMS. The data types normally match those used in the RDBMS table-creation statement.

4. Variables can be declared between the keywords IS and BEGIN. You must specify the variable name, its data type, and (optionally) an initial value.

**Procedures on MySQL databases**

**1) Counting number of tuples in table t**

```
DELIMITER //
CREATE PROCEDURE simpleproc (OUT param1 INT)
BEGIN
   SELECT COUNT(*) INTO param1 FROM t;
END//
DELIMITER ;

CALL simpleproc(@a);
SELECT @a;
```

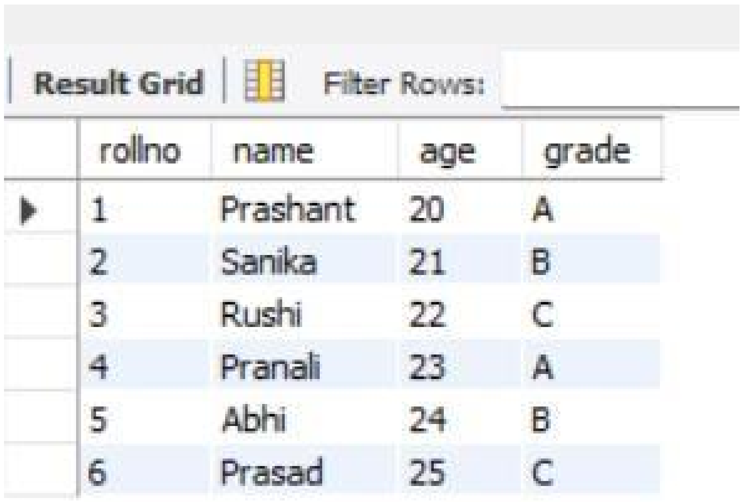


Result Grid

| | @a |
|---|---|
| ▶ | 6 |

**2. Display Information of Students**

```
DELIMITER $$
DROP PROCEDURE IF EXISTS getstudent $$CREATE
PROCEDURE getstudent()
BEGIN
   SELECT * FROM students;
END$$
DELIMITER ;

CALL getstudent();
```

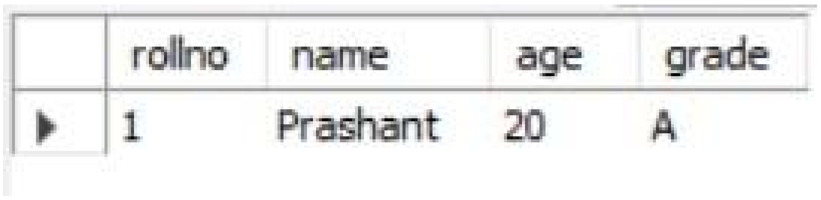


Result Grid   Filter Rows:

| | rollno | name | age | grade |
|---|---|---|---|---|
| ▶ | 1 | Prashant | 20 | A |
| | 2 | Sanika | 21 | B |
| | 3 | Rushi | 22 | C |
| | 4 | Pranali | 23 | A |
| | 5 | Abhi | 24 | B |
| | 6 | Prasad | 25 | C |

**3. Display Information of a Student by Roll Number**

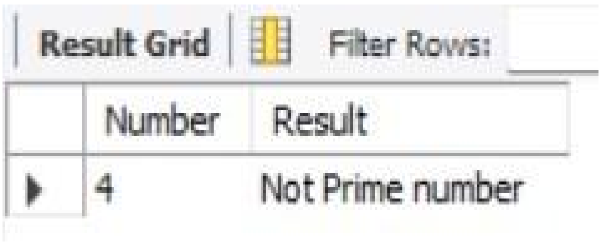
```
DELIMITER $$
DROP   PROCEDURE   IF   EXISTS   getstudent1
$$CREATE PROCEDURE getstudent1(IN rn INT)
BEGIN
   SELECT  *  FROM  students  WHERE  rollno  =  rn;
END$$
DELIMITER ;

CALL getstudent1(1);
```



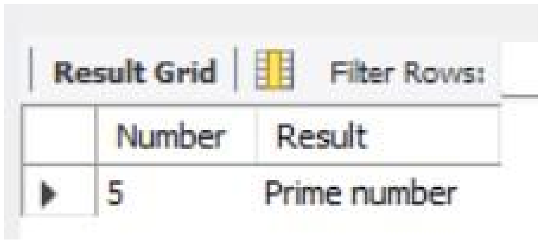
| | rollno | name | age | grade |
|---|---|---|---|---|
| ▶ | 1 | Prashant | 20 | A |

**4. Procedure to Find If a Number is Prime**

```
DELIMITER $$
DROP PROCEDURE IF EXISTS getprime_number
$$ CREATE PROCEDURE getprime_number(IN num
INT)BEGIN
   DECLARE x INT;
   DECLARE flag INT;SET
   x = 2;
   SET flag = 1;

   l1: WHILE x < num DO
      IF (num MOD x = 0) THEN
         SET flag = 0;
         LEAVE l1;
      ELSE
         SET x = x + 1;END
      IF;
   END WHILE;

   IF (flag = 1) THEN
      SELECT num AS Number, "Prime number" AS Result;ELSE
      SELECT num AS Number, "Not Prime number" AS Result;
   END IF;
END$$ DELIMITE
R ;

CALL getprime_number(5);
CALL getprime_number(4);
```



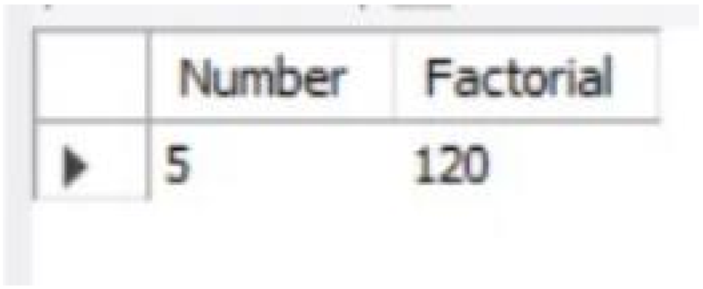
```
Number  Result
4       Not Prime number
```



```
Number  Result
5       Prime number
```

**5. Procedure to Find the Factorial of a Given Number**

```
DELIMITER //
DROP PROCEDURE IF EXISTS fact //
CREATE PROCEDURE fact(IN x INT)
BEGIN
   DECLARE result INT;
   DECLARE i INT;
   SET result = 1;SET
   i = 1;

   WHILE i <= x DO
      SET result = result * i;SET
      i = i + 1;
   END WHILE;

   SELECT x AS Number, result AS Factorial;END//
DELIMITER ;



call fact(5);
```



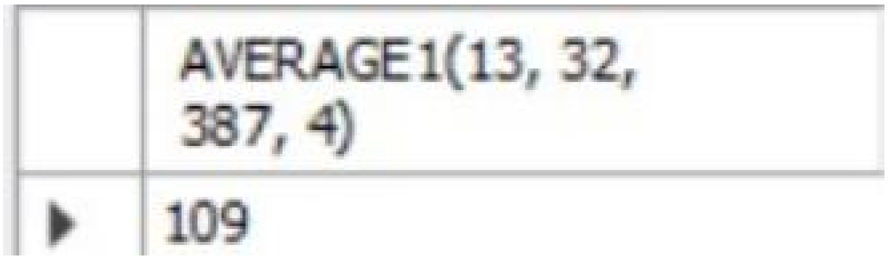
```
Number  Factorial
5       120
```

**Functions:** function always returns a value back to a calling block.
**Syntax: (Oracle database)**

```
Create [or replace] procedure procedurename
[[parameter 1[, parameter 2, …… , ]]]
 return data typeis
[constant / variable declaration]begin
executable statements
[Exception exception handling statements return returnvalue]End
[functionname];
```

**1. Function to Find the Average of Given Numbers**

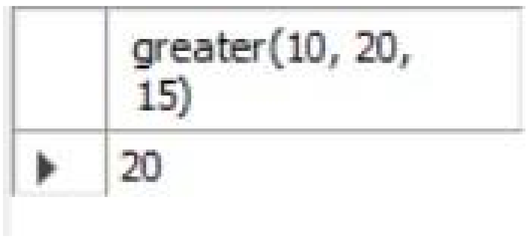
```
DELIMITER $$

DROP FUNCTION IF EXISTS AVERAGE1 $$

CREATE FUNCTION AVERAGE1(n1 INT, n2 INT, n3 INT, n4 INT)
RETURNS INT
DETERMINISTIC
BEGIN
   DECLARE avg INT;
   SET avg = (n1 + n2 + n3 + n4) / 4;
   RETURN avg;
END

$$ DELIMITER ;



SELECT AVERAGE1(13, 32, 387, 4);
```



```
AVERAGE1(13, 32,
387, 4)
109
```

**2. Function to Find the Greater of Three Numbers**

```
DELIMITER $$

DROP FUNCTION IF EXISTS greater $$

CREATE FUNCTION greater(n1 INT, n2 INT, n3 INT)
RETURNS INT
DETERMINISTIC  -- Specify that the function is deterministic
BEGIN
   DECLARE gr INT;

   IF (n1 > n2 AND n1 > n3) THEN
      SET gr = n1;
   ELSEIF (n2 > n1 AND n2 > n3) THEN
      SET gr = n2;
   ELSE
      SET gr = n3;END
   IF;

   RETURN gr;
END

$$ DELIMITER ;



SELECT greater(10, 20, 15);
```



```
greater(10, 20,
15)
20
```

**Outcome:** Students will able to implement Functions, Procedures using PL/SQL.