A *procedure* (often called a stored procedure) is a program that can be called to perform operations. A procedure can include both host language statements and SQL statements. Procedures in SQL provide the same benefits as procedures in a host language.

Types of parameters:

IN STANDS for input parameter which means input values will be passed in this parameter to AS400 program

OUT stands for output parameter which means output generated from AS400 side can be passed back in these kinds of parameters and received at the calling side.

INOUT stands for input output parameters which means AS400 program will receive input in this parameter and then whatever output is calculated/generated can be sent back in same parameter.

Basic SP

1, It can be created using create or replace procedure command.

2. Language will be SQL for SQL stored procedures which means entire logic is present in SQL procedure itself.

If it is external stored procedure which means it calls external RPGLE/CLP program language will be RPGLE or CLP. In this case we need to provide external program name.

3. Variables should be declared using declare statement.

4. Values of variables can be assigned using SET statements.

Example SET A = 1

SET B = A*C

Create or replace procedure proc_name
(parm1 IN DATATYPE,
PARM2 OUT DATATYPE)
LANGUAGE RPGLE
MODIFIES SQL DATA

1. Basic SQL stored procedure to take employ1ee number and city as input and update city for that employee number.

```
CREATE OR REPLACE PROCEDURE CITYUPD(
IN P_EMPNO NUMERIC(10) ,
IN P_CITY  CHAR(10))
LANGUAGE SQL
```

```
MODIFIES SQL DATA
         UPDATE EMPPF SET CITY     = P_CITY
             WHERE EMPNO = P_EMPNO
```

2. SQL SP using IF and end if

Syntax

```
IF CONDITION THEN
    ACTION
END IF
```

```
IF P_RATING = 'A'  THEN
  UPDATE EMPPF SET SALARY   = SALARY + 5000
              WHERE EMPNO = P_EMPNO ;
END IF ;

IF P_RATING = 'B' THEN
  UPDATE EMPPF SET SALARY   = SALARY + 2000
              WHERE EMPNO = P_EMPNO ;
END IF ;
```

3. SQLSP using case.

```
CASE P_RATING
 WHEN 'A' THEN
  UPDATE EMPPF SET SALARY   = SALARY + 5000
              WHERE EMPNO = P_EMPNO ;

 WHEN 'B' THEN
  UPDATE EMPPF SET SALARY   = SALARY + 2000
              WHERE EMPNO = P_EMPNO ;

 WHEN 'C' THEN
  UPDATE EMPPF SET SALARY   = SALARY + 1500
              WHERE EMPNO = P_EMPNO ;
  ELSE
  UPDATE EMPPF SET SALARY   = SALARY
              WHERE EMPNO = P_EMPNO ;
END CASE ;
```

SQLSP returning result set

```
CREATE or REPLACE PROCEDURE PRCEMPDET
(IN @EMPNO DECIMAL(6,0))
        DYNAMIC RESULT SETS 1
        LANGUAGE SQL
        Not Deterministic
        READS SQL DATA

Begin
  Declare SUMMARY cursor for select * from emPSALPF  WHERE EMPNO = @EMPNO;


 Open SUMMARY ;

Return  ;
End;
```

IF OUR SP is returning 2 result sets it will be defined with DYNAMIC RESULT SETS 2

SP returning result sets will only have cursor query and open cursor in them and fetch part will be based on front end setup.
This return of result sets will be done when front end makes and api call by using stored procedure and the api settings will get results in either json/xml. So all selected rows through SQL will be returned in JSON/XML format and front end will utilize that.

Stored procedure using FOR Loop

For statement will loop over rows of a result set.

When a FOR statement is executed, a cursor is implicitly declared such that for each iteration of the FOR-loop the next row is the result set if fetched. Looping continues until there are no rows left in the result set.

Syntax

For temp alias name as Cursorname Cursor for select query.

```
Create procedure exmpext ()
Language sql
Modifies sql data
Begin
FOR v AS cur1 CURSOR FOR
            SELECT firstnme, midinit, lastname FROM employee
   DO
     SET fullname = v.lastname || ',' || v.firstnme
                ||' ' || v.midinit;
     INSERT INTO tnames VALUES (fullname);
   END FOR;
  END ;
```

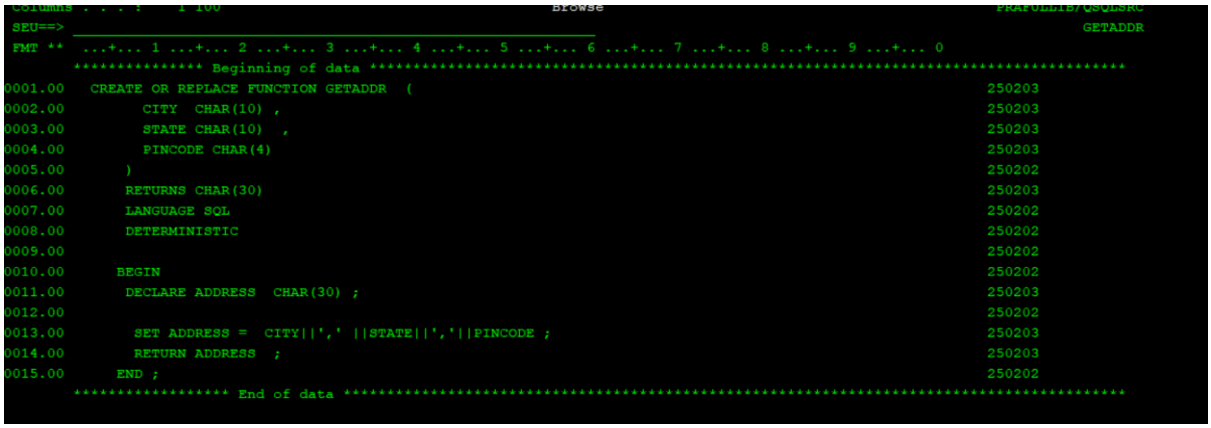
External proc calling CL /RPGLE program

```
CREATE OR REPLACE PROCEDURE PRCext
(IN PATH CHAR(100) ,
 OUT ERR_MESSAGE VARCHAR(500))
LANGUAGE RPGLE
SPECIFIC PRCEXT
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
EXTERNAL NAME PGMname
PARAMETER STYLE GENERAL WITH NULLS;
```

Sp ATTACHED to Service pgm(procedure)

```
CREATE OR REPLACE PROCEDURE PRCext
 (OUT PERROR VARCHAR(1))
LANGUAGE RPGLE
SPECIFIC PRCPUTVAL
NOT DETERMINISTIC
MODIFIES SQL DATA
CALLED ON NULL INPUT
EXTERNAL NAME 'extsrvpgm(VALIDATE)'
PARAMETER STYLE GENERAL WITH NULLS;
```

SQL UDF



```
SEU==>                                                                    GETADDR
FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 ...+... 9 ...+... 0
       ********** Beginning of data **************************************************************************
0001.00   CREATE OR REPLACE FUNCTION GETADDR  (                                                    250203
0002.00         CITY  CHAR(10) ,                                                                  250203
0003.00         STATE CHAR(10)  ,                                                                 250203
0004.00         PINCODE CHAR(4)                                                                   250203
0005.00       )                                                                                   250202
0006.00       RETURNS CHAR(30)                                                                    250203
0007.00       LANGUAGE SQL                                                                        250202
0008.00       DETERMINISTIC                                                                       250202
0009.00                                                                                           250202
0010.00      BEGIN                                                                                250202
0011.00       DECLARE ADDRESS  CHAR(30) ;                                                         250203
0012.00                                                                                           250202
0013.00        SET ADDRESS =  CITY||',' ||STATE||','||PINCODE ;                                   250203
0014.00        RETURN ADDRESS  ;                                                                  250203
0015.00      END ;                                                                                250202
       ********** End of data ****************************************************************************
```

Update query using this UDF:
UPDATE PRAFULLIB/CUSPF SET CUSTADDR = GETADDR(CUSCITY ,CUSSTATE , CUSSTPIN)

```
  WHEN 'DELETE' THEN
```

```
SELECT RESPONSE_HTTP_HEADER, RESPONSE_MESSAGE
    Into vRESPONSE_HDR, VResponse_Body
```

```
    FROM TABLE(QSYS2.HTTP_DELETE_VERBOSE(
    TRIM(vURL), pRequest, vRequestHeader)) as WS;
END CASE;

VALUES JSON_VALUE(VRESPONSE_HDR, '$.HTTP_STATUS_CODE'
                  DEFAULT 000 ON ERROR)
INTO VHTTPCODE;
IF VHTTPCODE <>  200       THEN

    SET VSTATUSCODE =  'E' ;

     SET vMsgText = 'Error response received.HTTP Status Code:'||
                  DIGITS(VHTTPCODE) ||'.' ;

      VALUES JSON_OBJECT(
      'StatusCode' VALUE 'E',                   JSO
      'StatusMessage' VALUE Trim(vMsgText) )
       Into pResponse;
  ELSE
     Set Presponse = VResponse_Body  ;
     SET VSTATUSCODE =  'S' ;
     SET vMsgText = ' '  ;
  END IF ;
```