SQL PL – Creative Uses

David Simpson
Themis Training
dsimpson@themisinc.com

November 2019
Session IH




SQL Procedure Language

SQL vs “External” Strengths and Limitations

Strengths

- Easy & fast
- Tools
- Little Server knowledge Required
- One step Creation
- Portable
- Do not run in WLM
- zIIP eligible DDF $ $ $

Limitations

- Data Structures
- No Performs
- No Includes / Copy Books
- Limited logic
- Performance
- Create Authorization
- Error logging
- Integration with existing Change Management tools

Coding a SQL PL Procedure

- An SQL procedure consists of:
  - CREATE PROCEDURE header
  - BEGIN statement
  - Body (SQL procedural statements and / or DB2 SQL statements)
  - END statement

- Comments within an SQL procedure:
  - -- for a single line comment
  - /* to start */ end multiple-lines comments

- Statements end with semicolon

A SQL PL Header

```
CREATE PROCEDURE SPA80
                 (IN  P_DNO  CHAR(3)
                 ,OUT P_CNT  SMALLINT
                 ,OUT P_SUMSAL DECIMAL(11,2)
                 ,OUT P_RETCODE INTEGER )
VERSION V1
```

Parameters

- Procedure Name
  - 128 byte max length
  - Unique within Schema / Collection
  - Schema / Collection ID will be supplied when create is deployed
- Parameters
  - 128 byte max length
  - Can be IN bound, OUT bound or INOUT (both directions)
  - Used to pass data between procedure and caller
  - Cannot specify a default value
- Versioning
  - 64 EBCDIC bytes max length
  - If using versioning do not use the default V1 naming convention

A SQL PL Body

The body consists of 5 parts:

- SQL variable declarations
- Condition names
- Cursors
- Condition Handlers
- Code…

A SQL PL Body

```
…
…
P1: BEGIN
      SET P_CNT = 0;
      SET P_SUMSAL = 0;
      SET P_RETCODE = 0;

      SELECT COUNT(*), SUM(SALARY)
        INTO P_CNT, P_SUMSAL
        FROM EMP
       WHERE DEPTNO = P_DNO;
END P1
```

Assign Values

Clear your outbound parms

SQL Procedure Statements

- DECLARE Statement
- Assignment Statement
- **CALL**, **GOTO**, **LEAVE**, **RETURN**
- **IF, CASE**, **WHILE**, **LOOP**, **REPEAT**, **ITERATE, FOR**
- Compound statement
- **GET DIAGNOSTICS** statement
- **SIGNAL, RESIGNAL** statements
- SQL Statements

- Note: Successful Execution of any SQL statement will set
  SQLCODE variable value to 0 and
  SQLSTATE variable value to ‘00000’.

SQL Variable Example

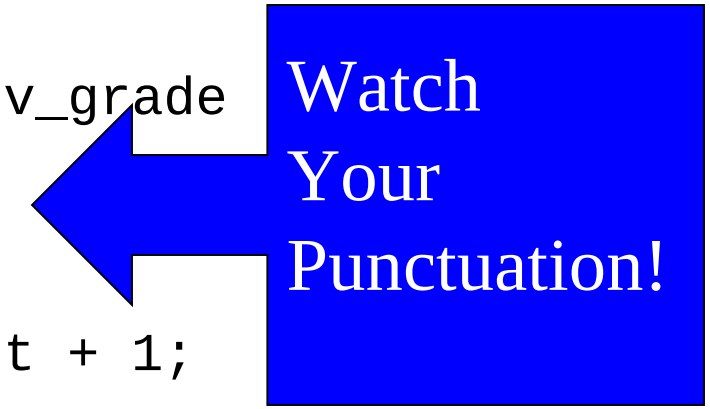
```
…
P1: BEGIN
    DECLARE SQLCODE INTEGER DEFAULT 0;
    DECLARE V_LAST_PAID_DATE DATE;
    DECLARE V1 CHAR(25) DEFAULT 'NOT PAID';
    DECLARE V2 INTEGER;
    DECLARE V3 DECIMAL(9,2);
    DECLARE V4 DECIMAL (9,2) DEFAULT 0;

    SET V2 = 1000;
    SET V3 = 500.00;
.
.
.
END P1
```

IF Statement

```
DECLARE v_grade CHAR(1);
DECLARE v_a_count INTEGER;
DECLARE v_b_count INTEGER;
DECLARE v_invalid_count INTEGER;
...
UPDATE STUDENT SET GRADE = v_grade
WHERE STUDENT_NO = var1;

IF v_grade = 'A' THEN
   set v_a_count = v_a_count + 1;
ELSEIF v_grade = 'B' THEN
   set v_b_count = v_b_count + 1;
ELSEIF . . . THEN  . . .
ELSE
   Set v_invalid_count = v_invalid_count + 1;
END IF;
```



IF Statement

```
IF v_grade = 'A' THEN
   set v_a_count = v_a_count + 1;
   set v_counter = v_counter + 1;
ELSEIF v_grade = 'B' THEN
   set v_b_count = v_b_count + 1;
   set v_counter = v_counter + 1;
ELSEIF . . . THEN  . . .
ELSE
   Set v_invalid_count = v_invalid_count + 1;
END IF;
```

CASE Statement

1. Simple CASE: Testing for value of variable

```
CASE v_grade
  WHEN 'A' THEN set v_a_count = v_a_count + 1;
                set v_counter = v_counter + 1;
  WHEN 'B' THEN set b_count = b_count + 1;
                set v_counter = v_counter + 1;
  ELSE set v_invalid_count = v_invalid_count + 1 ;
END CASE;
```

2. Searched CASE: Testing for TRUE condition

```
CASE
  WHEN v_edlevel < 12 THEN
       set v_no_diploma = v_no_diploma + 1;
  WHEN v_edlevel > 11 and v_edlevel < 16 THEN
       set v_high_school = v_high_school + 1;
END CASE;
```

LOOP, LEAVE & REPEAT

```
FETCH_LOOP: LOOP
      FETCH CURSOR1 INTO VAR1, VAR2, VAR3 ;
      IF SQLCODE = 100 THEN
            LEAVE FETCH_LOOP;
      END IF;
      <process values returned by cursor>
END LOOP;
```

```
REPEAT
      FETCH CURSOR1 INTO VAR1, VAR2, VAR3;
      IF SQLCODE = 100 THEN SET V_EOF = 'Y';
        ELSE
            <process values returned by cursor>
      END IF;
  UNTIL V_EOF = 'Y'
END REPEAT;
```

WHILE Statement

```
DECLARE SQLCODE             INTEGER DEFAULT 0;
DECLARE V_EOF       CHAR(1) DEFAULT 'N';
…
WHILE (V_EOF = 'N') DO
   FETCH CURSOR1 INTO VAR1, VAR2, VAR3;
   IF SQLCODE = 100 THEN
      SET V_EOF = 'Y';
   ELSE
      <process a row> . . . ;
   END IF;
END WHILE;
```

GOTO Statement

The GOTO transfers control to a labeled statement. The labeled statement and the GOTO statement must be in the same scope.

```
IF V_SERVICE < 10000 THEN
    GOTO EXIT_RTN ;
END IF ;
…;
…;
EXIT_RTN:
  BEGIN
   SET P_RETURN_CODE = V_SERVICE ;
  END;
```

ITERATE Statement

The ITERATE statement causes the flow of control to return to the beginning of a labeled loop.

```
WHILE_ROUTINE :
   WHILE (MORE_RESULT = 0) DO
         FETCH CURSOR1 INTO VAR1, VAR2, VAR3;
         SET MORE_RESULT = SQLCODE;
         IF VAR3 < 0 THEN
               ITERATE WHILE_ROUTINE;
         END IF ;
         . . . ;
         . . . ;
   END WHILE;
```

Exception Handling

Capturing Exceptions

```
P1: BEGIN
      DECLARE SQLCODE  INTEGER DEFAULT 0;
      DECLARE SQLSTATE CHAR(5) DEFAULT '00000';

      ...
      FETCH CURSOR1 INTO V1, V2;
      IF SQLCODE = 100 THEN
            SET EOF = 'Y';
```

Unhandled Errors

```
CREATE PROCEDURE SPERR (OUT P_RETCODE INTEGER)
       VERSION VERSION1
P1: BEGIN
       DECLARE SQLCODE   INTEGER DEFAULT 0;
       DECLARE V1        DATE;

       SET P_RETCODE = 0;
       SELECT DATE('2019-12-32')
         INTO V1
         FROM SYSIBM.SYSDUMMY1;

       SET P_RETCODE = SQLCODE;
END P1
```

Handlers


DECLARE
EXIT
CONTINUE
HANDLER FOR
SQLEXCEPTION
SQLWARNING
NOT FOUND
SQLSTATE ‘value’

Uses for SQL PL on Db2

- Stored Procedures
- User Defined Functions (UDFs)
- Triggers (Db2 12)

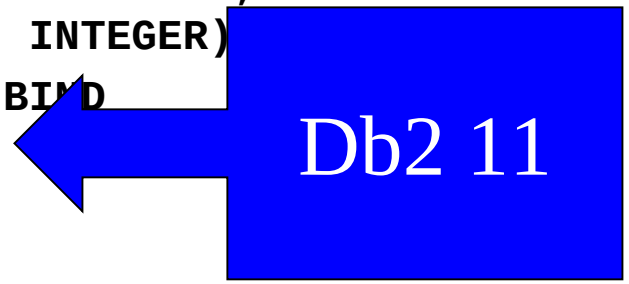
Error Handling Stored Procedure

```
CREATE PROCEDURE LOG_ERROR
    ( IN  P_PGMNAME   CHAR(30),
      IN  P_SQLCODE INTEGER,
      IN  P_MESSAGE   CHAR(240),
      IN  P_LINENUM   INTEGER ,
     OUT  P_STATUS    INTEGER)
 VERSION V1 VALIDATE BIND
 AUTONOMOUS

P1: BEGIN
  DECLARE SQLCODE INTEGER DEFAULT 0;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING
    BEGIN
       SET P_STATUS = SQLCODE;
    END;

  INSERT INTO THEMIS.LOG_ERROR
   (LOG_TIME, LOG_PGM, LOG_CODE, LOG_MSG, LOG_LINE)
   VALUES (CURRENT TIMESTAMP,P_PGMNAME,P_SQLCODE,P_MESSAGE,P_LINENUM);


   COMMIT;
END P1
```



Calling Error Handling

SQLCODE < 0

SQLCODE > 0
but NOT 100

```
DECLARE EXIT HANDLER FOR SQLEXCEPTION, SQLWARNING
   BEGIN
      DECLARE V_MESSAGE VARCHAR(240) DEFAULT '';
      DECLARE V_LINE INTEGER DEFAULT 0;
      DECLARE V_STATUS CHAR(1);

      SET P_SQLCODE = SQLCODE;
      GET STACKED DIAGNOSTICS CONDITION 1
           V_MESSAGE = MESSAGE_TEXT,
           V_LINE = DB2_LINE_NUMBER;

      CALL LOG_ERROR('SP5N00',P_SQLCODE,V_MESSAGE,V_LINE,V_STATUS);
   END;
```

Functions

Db2 Built-in Functions (BIFs)

Scalar Functions:

```
SELECT      SUBSTR(LASTNAME,1,1),
            YEAR(HIREDATE),
            COALESCE(SALARY,0)
  FROM      EMP
WHERE…
```

Column (Aggregate) Functions:

```
SELECT      SUM(SALARY),
            COUNT(*)
  FROM      EMP
WHERE…
```

Db2 Built-in Functions (BIFs)

Table Functions:

```
SELECT T.*
FROM PATIENT P,
XMLTABLE (XMLNAMESPACES(DEFAULT 'http://xyz.com/aSchema'),
                '$p/patient' PASSING P.PATIENT_XML as "p"
  COLUMNS "PATIENTNAME" VARCHAR(30)  PATH 'name',
          "SERVDATE"    DATE         PATH 'service/sdate',
          "REASON"      VARCHAR(30)  PATH 'service/reason',
          "DESCRIPTION" VARCHAR(30)  PATH 'service/descrip',
          "COST"        DECIMAL(7,2) PATH 'service/cost',
          "COPAY"       DECIMAL(7,2) PATH 'service/copay' ) as T;
```

User Defined Functions (UDFs)

GUIDE SHARE EUROPE UK REGION

Scalar Functions:

```
SELECT ADD_BUSINESS_DAYS(DUE_DATE, 5)
  FROM PROJECT
 WHERE …
```

A UDF that adds a number of days to a date but excludes weekends and company defined holidays.

UDF Example 1

```
CREATE FUNCTION THEMIS.ADD_BUSINESS_DAYS
   (A_DAY_IN DATE, A_NUM_DAYS SMALLINT)
     RETURNS    DATE
    DETERMINISTIC
    NO EXTERNAL ACTION
    VERSION V1
BEGIN
 --Declare variables
 DECLARE V_RETURN DATE;
 DECLARE V_CNT SMALLINT DEFAULT 1;
 DECLARE V_MONTH  INTEGER;
 DECLARE V_DAY  CHAR(4);
 DECLARE V_HOLIDAY CHAR(1);
 DECLARE V_DAYOFWEEK INTEGER;
 DECLARE V_INCREMENT SMALLINT DEFAULT 1;

 --Declare conditions
 DECLARE MONDAY_HOLIDAY CONDITION FOR SQLSTATE '75001';
```

Input arguments

Variables for the program

UDF Example 1

```
--Declare handlers
   DECLARE EXIT HANDLER FOR SQLEXCEPTION,SQLWARNING
    BEGIN
       DECLARE V_MESSAGE VARCHAR(240) DEFAULT '';
       DECLARE V_LINE INTEGER DEFAULT 0;
       DECLARE V_STATUS CHAR(1);
       DECLARE V_SQLCODE INTEGER;

       GET STACKED DIAGNOSTICS CONDITION 1
         V_MESSAGE = MESSAGE_TEXT,
         V_LINE = DB2_LINE_NUMBER,
         V_SQLCODE = DB2_RETURNED_SQLCODE;

       CALL THEMIS.SPERROR('ADD_BUSINESS_DAYS',
                           V_SQLCODE, V_MESSAGE,
                           V_LINE, V_STATUS);
       RESIGNAL;
    END;
```

UDF Example 1

```
DECLARE CONTINUE HANDLER FOR MONDAY_HOLIDAY
   BEGIN
      SET V_HOLIDAY = 'N';
      CASE
        WHEN V_DAY BETWEEN '0115' AND '0121' THEN    -- MLK DAY
           SET V_HOLIDAY = 'Y';
        WHEN V_DAY BETWEEN '0215' AND '0221' THEN    -- PRES DAY
           SET V_HOLIDAY = 'Y';
        WHEN V_DAY BETWEEN '0524' AND '0531' THEN    -- MEMORIAL DAY
           SET V_HOLIDAY = 'Y';
        WHEN V_DAY BETWEEN '0901' AND '0907' THEN    -- LABOR DAY
           SET V_HOLIDAY = 'Y';
        WHEN V_DAY BETWEEN '1008' AND '1014' THEN    -- COLUMBUS DAY
           SET V_HOLIDAY = 'Y';
        ELSE
           SET V_HOLIDAY = 'N';
      END CASE;
   END;
```

UDF Example 1

```
-- Get out if parms are bad.
  IF A_NUM_DAYS IS NULL OR
    A_DAY_IN IS NULL
    THEN SIGNAL SQLSTATE '75003'
      SET MESSAGE_TEXT = 'Arguments may not be null';
  ELSEIF A_NUM_DAYS NOT BETWEEN -720 AND 720 THEN
    SIGNAL SQLSTATE '75002'
      SET MESSAGE_TEXT = 'Parm 2 must be between -720 and +720’;
  END IF;

 IF A_NUM_DAYS < 0 THEN
    SET V_INCREMENT = -1;
 END IF;

 -- Keep looking at the next day until you find the one you want
 SET V_RETURN = A_DAY_IN;
 IF A_NUM_DAYS = 0 THEN
    GOTO EXIT;
 END IF;
```

Throw exception for bad input

Are we moving forward or backward?

Zero means we are already done

UDF Example 1

```
L1: LOOP
    -- add (or subtract) a day
    SET V_RETURN = V_RETURN + V_INCREMENT DAYS;

    SET V_MONTH = MONTH(V_RETURN);
    SET V_DAY =   RIGHT(DIGITS(V_MONTH),2)    -- build month/day
       || RIGHT(DIGITS(DAY(V_RETURN)),2);
    SET V_DAYOFWEEK = DAYOFWEEK(V_RETURN);

    IF V_DAYOFWEEK IN (1,7) THEN    -- Weekends don't count
       ITERATE L1;
    END IF;
    IF V_DAY IN ('0101','0704','1111','1225') THEN  -- Fixed Holidays
       ITERATE L1;
    END IF;
```

UDF Example 1

```
IF V_DAYOFWEEK = 2 THEN   -- mondays
     IF V_MONTH IN (1,2,5,9,10) THEN  -- check bank mondays
        SIGNAL MONDAY_HOLIDAY;
        IF V_HOLIDAY = 'Y' THEN
           ITERATE L1;
        END IF;
     END IF;
     --FIXED HOLIDAY WAS WEEKEND TRANSFER TO MONDAY
     IF V_DAY IN ('0102','0103','0705','1112','1226') THEN
        ITERATE L1;
     END IF;
   END IF;
```

UDF Example 1

```
 -- FIXED HOLIDAY WAS SATURDAY transfer to friday
    IF V_DAYOFWEEK = 6 AND
       V_DAY IN ('0703','1110','1224') THEN
       ITERATE L1;
    END IF;
       IF V_DAY BETWEEN '1122' AND '1128' AND
          V_DAYOFWEEK = 5 THEN         -- Thanksgiving Day (USA)
          ITERATE L1;
       END IF;
    IF V_CNT >= ABS(A_NUM_DAYS) THEN    -- WE'RE DONE
        LEAVE L1;
       END IF;
    SET V_CNT = V_CNT + 1;
 END LOOP;

EXIT:
  RETURN V_RETURN;
END
```

If I make it all the way here then the day counts!

Invoking the UDF

```
SELECT ADD_BUSINESS_DAYS(DATE('2018-12-21'),2)
   FROM SYSIBM.SYSDUMMY1;
```

This date is the Friday before Christmas

Saturday doesn't count

Sunday doesn't count

Monday counts (that's day 1)

Christmas Day (Tuesday) doesn't count

Query result is: 2018-12-26

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
DECEMBER 2018						
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21 0	22 X
23 X	24 1	25 X	26 2	27	28	29
30	31					

UDF Example 2

```
CREATE FUNCTION THEMIS.INITCAP (A_INSTR VARCHAR(32000))
     RETURNS    VARCHAR(32000)
DETERMINISTIC
NO EXTERNAL ACTION
VERSION V1

BEGIN
 DECLARE V_DONE CHAR(1) DEFAULT 'N';
 DECLARE V_NEXTCAP CHAR(1) DEFAULT 'Y';
 DECLARE V_RETURN VARCHAR(32000) DEFAULT '';
 DECLARE V_CNT SMALLINT DEFAULT 1;
 REPEAT
    IF V_NEXTCAP = 'Y' THEN
       SET V_RETURN = V_RETURN || UPPER(SUBSTR(A_INSTR,V_CNT,1));
    ELSE
       SET V_RETURN = V_RETURN || LOWER(SUBSTR(A_INSTR,V_CNT,1));
    END IF;
```

UDF Example 2

```
    IF SUBSTR(A_INSTR,V_CNT,1) = ' ' THEN
       SET V_NEXTCAP = 'Y';
    ELSE
       SET V_NEXTCAP = 'N';
    END IF;

    IF V_CNT >= LENGTH(RTRIM(A_INSTR)) THEN
       SET V_DONE = 'Y';
    ELSE
       SET V_CNT = V_CNT + 1;
    END IF;
  UNTIL V_DONE = 'Y'
 END REPEAT;

 RETURN V_RETURN;

END
```

UDF Example 3

```
CREATE FUNCTION THEMIS.INITCAP_BETTER (A_INSTR VARCHAR(32000))
     RETURNS    VARCHAR(32000)
DETERMINISTIC
NO EXTERNAL ACTION

BEGIN
 DECLARE V_RETURN VARCHAR(32000) DEFAULT '';
 DECLARE V_INSTR VARCHAR(32000) DEFAULT '';
 DECLARE V_POS SMALLINT DEFAULT 2;
 DECLARE V_POSN SMALLINT DEFAULT 0;
 DECLARE V_POSO SMALLINT DEFAULT 0;

SET V_INSTR = RTRIM(A_INSTR);
SET V_POS = LOCATE(' ', V_INSTR, OCTETS);
```

UDF Example 3

```
IF V_POS = 0 THEN   -- ONLY ONE WORD PRESENT
   SET V_RETURN = UPPER(SUBSTR(V_INSTR,1,1))
                || LOWER(SUBSTR(V_INSTR,2));
   GOTO END;
END IF;

SET V_RETURN =  UPPER(SUBSTR(V_INSTR,1,1))
             || LOWER(SUBSTR(V_INSTR,2,V_POS-1))
             || UPPER(SUBSTR(V_INSTR,V_POS+1,1));

SET V_POSO = V_POS;
SET V_POS = V_POS + 1;  -- Don't need to look at the thing we
                        --  just uppercased above. This is
                        --  significant if it's another space
```

Deal with the easy case and get out

UDF Example 3

```
L1: LOOP  -- This will happen once per word
   SET V_POSN = LOCATE(' ',SUBSTR(V_INSTR,V_POS+1));
   SET V_POS = V_POS + V_POSN;

   IF V_POSN = 0 THEN -- We are done. Tack on the remainder
      SET V_RETURN = V_RETURN
           || LOWER(SUBSTR(V_INSTR,LENGTH(V_RETURN)+1));
      LEAVE L1;
   ELSEIF V_POS = V_POSO + 1 THEN
      SET V_RETURN = V_RETURN
                  || UPPER(SUBSTR(V_INSTR,V_POS+1,1));
   ELSE
      SET V_RETURN = V_RETURN
                 || LOWER(SUBSTR(V_INSTR,V_POSO+2,V_POS-V_POSO-2))
                 || ' ' || UPPER(SUBSTR(V_INSTR,V_POS+1,1));
   END IF;
   SET V_POSO = V_POS;
END LOOP;
```

Find the next space

UDF Example 3

```
 END:
   RETURN RTRIM(V_RETURN);


END
```

Result

```
1
2
3 SELECT PROJNO, PROJNAME,
4        THEMIS.INITCAP_BETTER(PROJNAME) AS INITCAP
5   FROM THEMIS91.PROJ
6  WHERE PROJNO LIKE 'M%';
7
```

Properties | SQL Results

	PROJNO	PROJNAME	INITCAP
1	MA2100	WELD LINE AUTOMATION	Weld Line Automation
2	MA2110	W L PROGRAMMING	W L Programming
3	MA2111	W L PROGRAM DESIGN	W L Program Design
4	MA2112	W L ROBOT DESIGN	W L Robot Design
5	MA2113	W L PROD CONT PROGS	W L Prod Cont Progs

Considerations

- Am I worried about performance?.... Yes
- How often will it be called?
- This one will be more expensive with larger intervals (second parm)
- How often will it be called?

Triggers

“Advanced” Trigger

```
CREATE TRIGGER DAVE81.TRGEMP01
AFTER UPDATE ON DAVE81.EMP
REFERENCING  OLD AS OE FOR EACH ROW

P1: BEGIN
   DECLARE EXIT HANDLER FOR SQLEXCEPTION
     BEGIN
        DECLARE V_SQLCODE INTEGER;   DECLARE V_MESSAGE VARCHAR(240);
        DECLARE V_LINE INTEGER;      DECLARE V_STATUS CHAR(1);

        GET STACKED DIAGNOSTICS CONDITION 1
           V_SQLCODE = DB2_RETURNED_SQLCODE,
           V_MESSAGE = MESSAGE_TEXT,
           V_LINE = DB2_LINE_NUMBER;

        CALL THEMIS.LOG_ERROR('TRGEMP01',V_SQLCODE, V_MESSAGE,
                              V_LINE, V_STATUS);
        RESIGNAL;
     END;
```

“Advanced” Trigger

```
   IF OE.EMPNO > '499999' THEN
         INSERT INTO OLD_EMP_HI (EMPNO,FIRSTNME, …
          VALUES (OE.EMPNO, OE.FIRSTNME,…
   ELSE
         INSERT INTO OLD_EMP_LOW (EMPNO,FIRSTNME, …
          VALUES (OE.EMPNO, OE.FIRSTNME,…
   END IF;

END P1
```

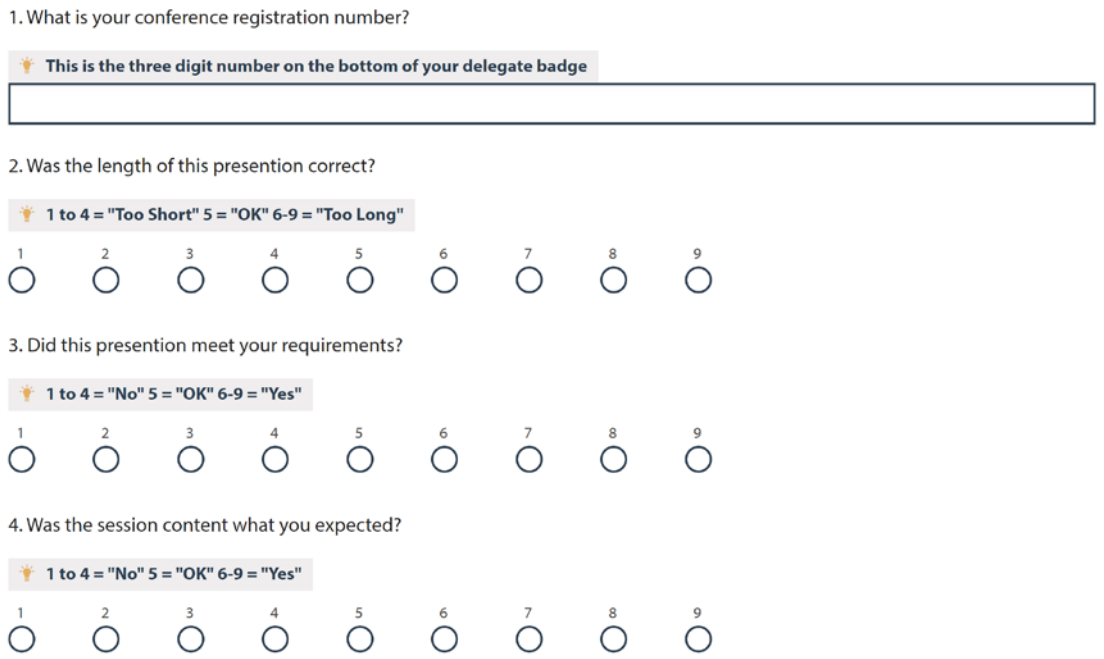
SQL PL – Creative Uses

David Simpson
Themis Education
dsimpson@themisinc.com
www.themisinc.com

@ThemisDave
@ThemisTraining

Please submit your session feedback!

- Do it online at http://conferences.gse.org.uk/2019/feedback/nn
- This session is IH



Place your custom session QR code here. Please remove the border and text beforehand.