Welcome to Lyon, France and the 2026 Common Europe Congress

Bienvenue à Lyon, en France, et au Congrès de Common Europe 2026

Agenda

What will we cover today ?

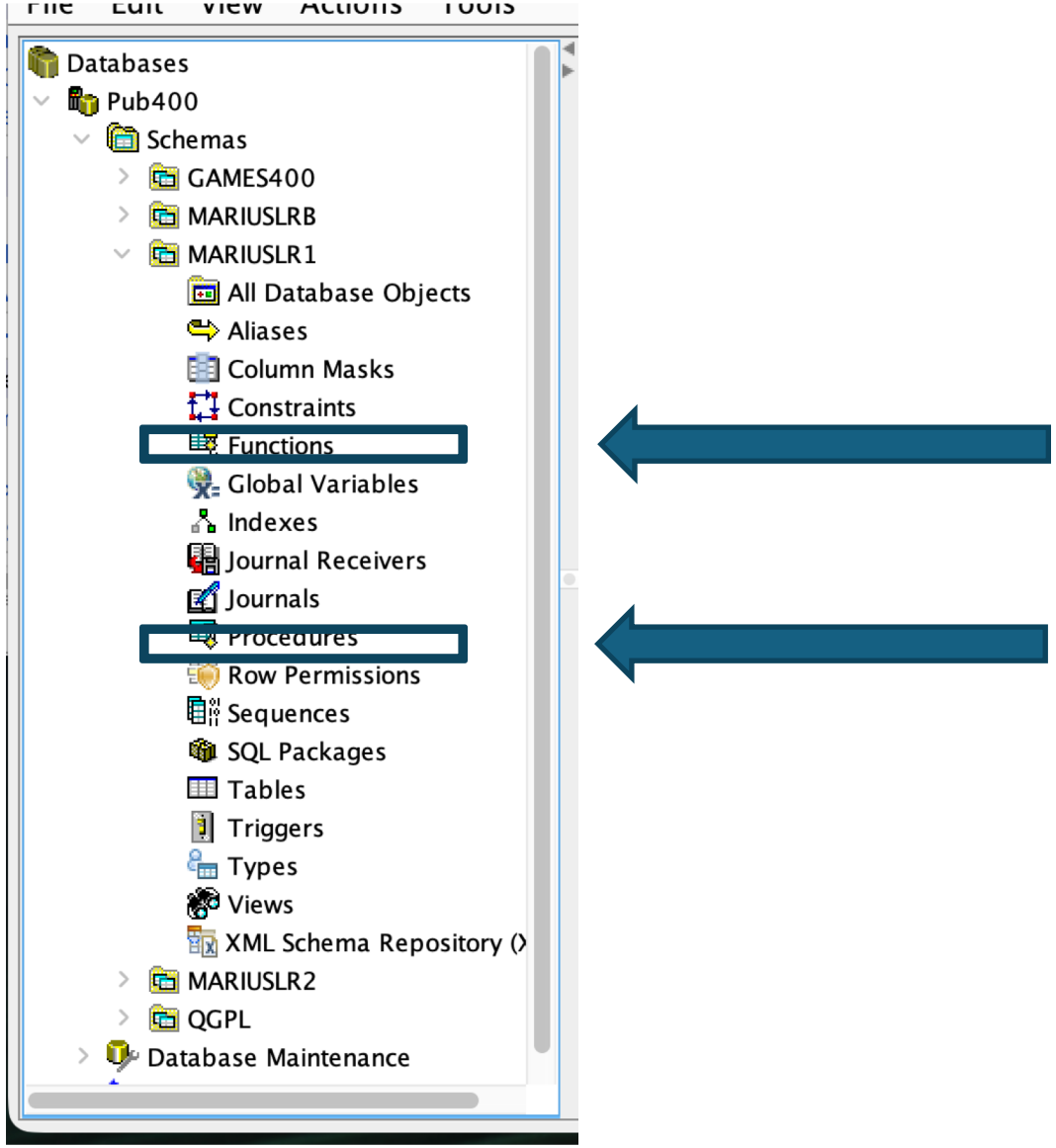
- What are DB2 for i Artefacts and why they matter.
- SQL Functions – User Defined Scalar and Table Functions
- SQL Procedures
- Modernization and Security
- Cool Things

Who Am i ?

Marius le Roux

IBM i Consultant

DB2 Artefacts



These are the ones we will cover today

SQL Procedural Language (SQL PL)

- SQL is also a programming language!

- Use it to write:
  - Procedures
  - Scalar functions
  - Table functions
  - Triggers
  - Compound (dynamic) statements

- Combine procedures, functions, triggers, and statements for powerful results

SQL PL Programming Key Concepts

- Variables
- Error handling
- Control statements:
  - IF
  - CASE
  - LOOP
- CALL statement
- Nested code blocks
- SQL statements

Variables

- DECLARE defines variables in the routine

- By default, all variables are nullable and initialized to the null value
  - Can also specify a default value

```
DECLARE v_midinit, v_edlevel CHAR(1);
DECLARE v_ordQuantity INT DEFAULT 0;
DECLARE v_enddate DATE DEFAULT CURRENT DATE;
DECLARE c_maxlength INT CONSTANT 128;
```

IF Statements

- Basic IF and IF / ELSE:

```
IF ref_error = 1 THEN
   SET o_error = 'NOT FOUND';
END IF;
```

```
IF ref_error = 1 THEN
   SET o_error = 'NOT FOUND';
ELSE
   SET o_error = 'FOUND';
END IF;
```

LOOP, LEAVE, and ITERATE Statements

- LOOP

No loop condition! Must use LEAVE to exit the loop

```
the_loop: LOOP
   CALL work_to_do(all_done);
   IF all_done = 1 THEN
      LEAVE the_loop;
   END IF;
END LOOP the_loop;
```

- LEAVE – exit the loop

- ITERATE – return to the top of the loop

WHILE and REPEAT Statements

- SQL PL has several loop types which differ in how the loop condition is checked

```
SET all_done = 0;
the_loop: WHILE all_done = 0 DO
   CALL work_to_do(all_done, hit_error);
   IF hit_error = 1 THEN
      LEAVE the_loop;
   END IF;
END WHILE the_loop;
```

Loop condition checked at start. Loop might not execute.

```
the_loop: REPEAT
   CALL work_to_do(all_done, hit_error);
   IF hit_error = 1 THEN
      LEAVE the_loop;
   END IF;
UNTIL all_done = 1
END REPEAT the_loop;
```

Functionally equivalent to a do-while loop. Loop will execute at least once.

What are the different types of Functions?

- There are two types of functions:

1. Scalar functions
   - Return single value
   - Invoked from almost any SQL statement
   - Good for encapsulating complex calculations and logic
   - Can be interfaced with RPGLE / Java / CLLE Programs

2. Table functions
   - Return set of rows (1 to Many)
   - Invoked from a query's FROM clause
   - Good for encapsulating complex calculations or non-traditional data
   - Can also be interfaced with RPGLE / Java / CLLE Programs

Creating Scalar Functions (SQL Variant)

- Functions are created using CREATE FUNCTION:

```
CREATE FUNCTION DateOfCYYMMDD ( i_cyymmdd CHAR(7) )
  RETURNS DATE
BEGIN
  DECLARE v_date CHAR(10);
  IF SUBSTR(i_cyymmdd,1,1) = '0' THEN
    SET v_date ='19' CONCAT SUBSTR(i_cyymmdd,2,2) CONCAT '-' CONCAT
      SUBSTR(i_cyymmdd,4,2) CONCAT '-' CONCAT SUBSTR(i_cyymmdd,6,2);
  ELSE
    SET v_date ='20' CONCAT SUBSTR(i_cyymmdd,2,2) CONCAT '-' CONCAT
      SUBSTR(i_cyymmdd,4,2) CONCAT '-' CONCAT SUBSTR(i_cyymmdd,6,2);
  END IF;
  RETURN ( DATE(v_date) );
END;
```

Creating Scalar Functions (RPGLE Variant)

- Functions are created using CREATE FUNCTION:

```
CREATE OR REPLACE FUNCTION GetItemUDF
        (
                in_ItemNbr VARCHAR(7),
                in_inItemLoc VARCHAR(5)
        )
    returns varchar(3)
    language rpgle
    external name ITEMCHECK
    parameter style sql
    not fenced
    program type main;
```

Creating Scalar Functions (RPGLE Variant , continued)

- RPGLE Sample code:

```
**free
    ctl-opt main(Main)
    dftactgrp(*no)
    actgrp(*caller)
    option(*srcstmt:*nodebugio);

// ------------------------------------------------------------------//
// Program.....: ITEMCHECK //
// Purpose.....: Determine if Item Exists in File UDF example. //
// ------------------------------------------------------------------ //

dcl-pr Main extpgm('ITEMCHECK');
  inItemNbr      varchar(2)        const;
  inItemLoc      varchar(5)        const;
  outExists      varchar(3);
  int_ItemNbr    int(5)         const;
  int_ItemLoc    int(5)         const;
  int_outExists  int(5);
  sqlStErr       char(5);
  fctName        varchar(517);
  spcName        varchar(128);
  msgText        varchar(1000);
end-pr;
```

SQL Specific Variables for Parameter Style SQL for State , Message Text and Function Name

Creating Scalar Functions (RPGLE Variant , continued)

- RPGLE Sample code:

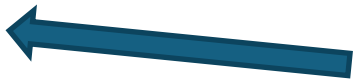
```
dcl-s notfound ind inz(*on);

// just for demo....
if inItemNbr = '11' and inItemLoc = 'BIN11';
   //set Example Return output here.
   outExists = 'YES';
   notfound = *off;
endif;

// Do some work here to see if item exists or not
if notFound;
  outExists = 'NO';
  sqlStErr = '59999';
  msgText  = 'Item not found';
endif;

return;

end-proc;
```



Here you can add error handling that gets returned to the SQL Client Caller

Creating Table Functions (SQL Variant)

- A table function returns a table
  - Can return an empty table (use this to your advantage)

```
CREATE FUNCTION BestSales (i_BonusThreshold DECIMAL(11,2))
            RETURNS TABLE (Empid INT,
               LastName VARCHAR(30),
               FirstName VARCHAR(30))
READS SQL DATA
NO EXTERNAL ACTION
RETURN
  SELECT id, lname, fname FROM custsales
    GROUP BY id, lname, fname
      HAVING SUM(sales) > i_BonusThreshold;
```

- There must be exactly one RETURN statement and it must be a full select

Creating Table Functions (RPGLE Variant)

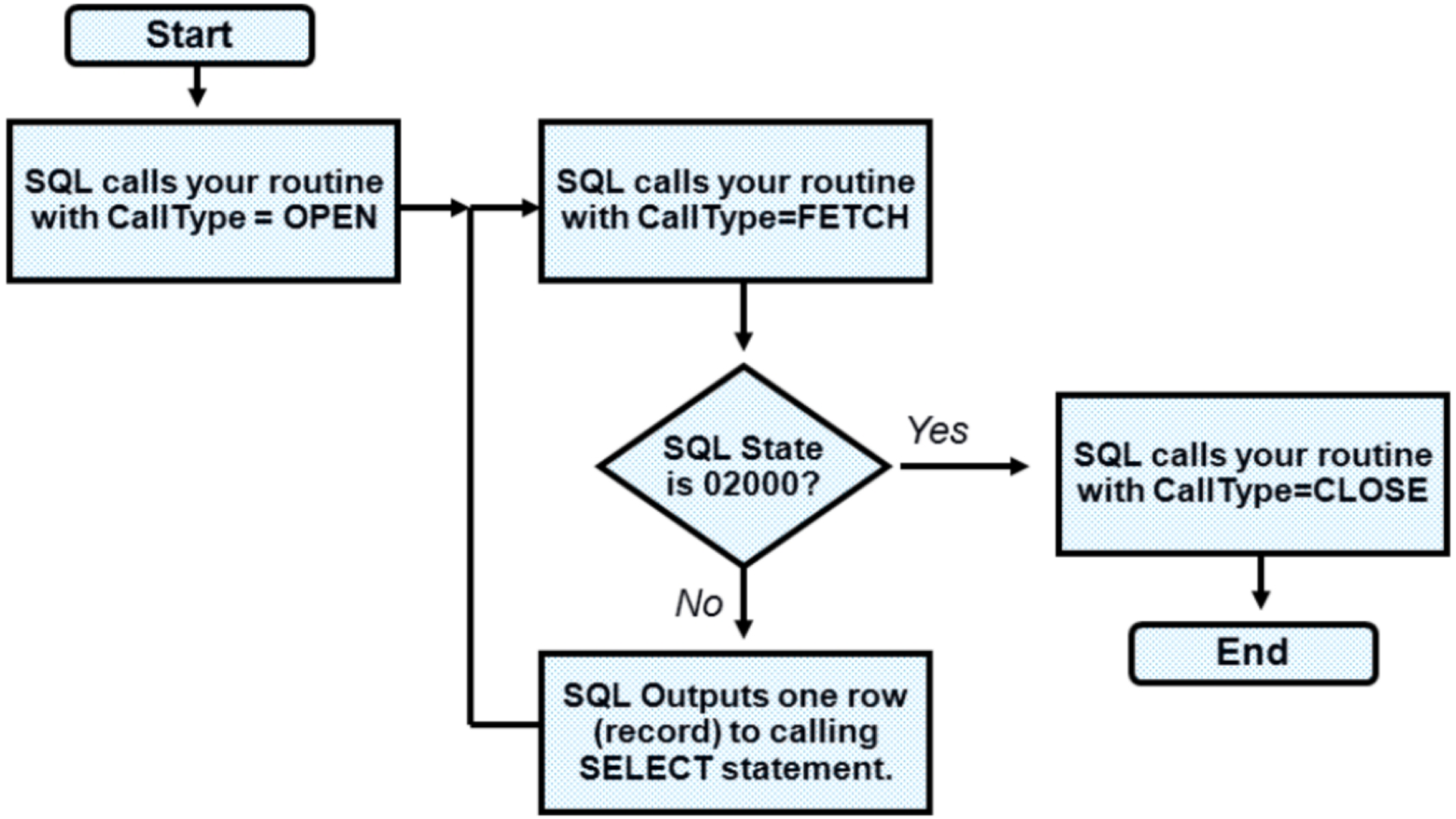
- A table function returns a table
  - RPGLE / CLLE / Java can also be used to retrieve data in a UDTF Form

```
CREATE OR REPLACE FUNCTION udtfexample1 ()
   RETURNS TABLE (
      OUT_TIMESTAMP CHAR(26)
   )
   LANGUAGE RPGLE
   PARAMETER STYLE DB2SQL
   NOT DETERMINISTIC
   NO SQL
   CALLED ON NULL INPUT
   NO DBINFO
   NO EXTERNAL ACTION
   NOT FENCED
   NO FINAL CALL
   DISALLOW PARALLEL
   SCRATCHPAD
   EXTERNAL NAME LIBRARY.SAMPLEUDTF
   CARDINALITY 1
```

- Notice the syntax for any Language that isn't same as SQL Variant

UDTF's "Cycle Logic" (for HLL)

Figure 3: The flow of events in a UDTF



Credit to Scott Klement's work on UDTFs:
https://www.scottklement.com/udtf/UDTFs%20for%20the%20Win.html

Invoking Table Functions

- A table function is invoked in the FROM clause

```
SELECT LastName, Empid
  FROM TABLE(BestSales(100000));
```

- Use named parameters to document parameter values

```
SELECT LastName, Empid
  FROM TABLE(BestSales(i_BonusThreshold => 100000));
```

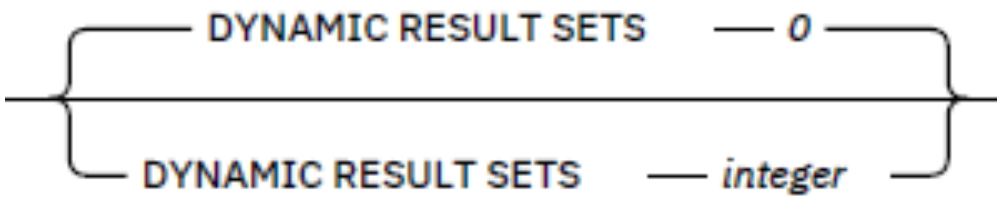
Creating Procedures

- Procedures are created using CREATE PROCEDURE:

```
CREATE OR REPLACE PROCEDURE increase_salary
   ( IN i_empno VARCHAR(6),
     IN i_increase DECIMAL(9,2),
     OUT o_salary DECIMAL(9,2) )
SET OPTION
            DBGVIEW=*SOURCE,
            USRPRF=*OWNER
BEGIN
   UPDATE employee SET salary = salary + i_increase
      WHERE empno = i_empno;
   SELECT salary INTO o_salary FROM employee
      WHERE empno = i_empno;
END;
```

Create Procedure – Result Sets

- A result set is an SQL cursor, with rows & columns defined by the procedure
- DYNAMIC RESULT SETS specifies the number of result sets that can be returned from procedure
  - Consumed by ODBC, JDBC, .NET, & CLI clients
  - Consumed by Embedded SQL programs and SQL routines



Creating Procedures

- Result Set Procedures are created using CREATE PROCEDURE:

```
CREATE OR REPLACE PROCEDURE getResultSetExample1 ()
       LANGUAGE SQL
       DYNAMIC RESULT SETS 1
SET OPTION
            DBGVIEW=*SOURCE,
            USRPRF=*OWNER
BEGIN
   DECLARE C1 CURSOR FOR
      SELECT *
         FROM QIWS.QCUSTCDT;
   OPEN C1;
   RETURN;
END;
```

- Parameters can be input, output, or both input and output also
- Return returns as many as Result sets defined

Creating Procedures

- Result Set Procedures are created using CREATE PROCEDURE:

```
CREATE OR REPLACE PROCEDURE getResultSetExample2 ()
        LANGUAGE Sql
        DYNAMIC RESULT SETS 1
        SET OPTION DBGVIEW = *SOURCE,
                USRPRF = *OWNER
BEGIN
    DECLARE customerList CURSOR WITH RETURN TO CLIENT FOR
        SELECT CUSNUM,
              LSTNAM,
              INIT,
              STREET
            FROM QIWS.QCUSTCDT
            FOR READ ONLY;
    OPEN customerList;
END;
```

- Result Sets then named

SQL Result Set Consumption

- Three statements are required to consume a result set created by a procedure

1. DECLARE RESULT_SET_LOCATOR VARYING
   - Defines a variable which identifies a result set
2. ASSOCIATE LOCATOR
   - Ties the locator to a procedure result set
3. ALLOCATE CURSOR
   - Connects the SQL cursor to the result set

Followed by FETCH (1 – many) and CLOSE statements

Result Set Consumption from RPG

```
DCL-S PROJ_LOC SQLTYPE(RESULT_SET_LOCATOR);

EXEC SQL CALL GetProjects(:projdept);

EXEC SQL ASSOCIATE LOCATOR (:PROJ_LOC)
        WITH PROCEDURE Get_Projects;

EXEC SQL ALLOCATE PROJ_CSR CURSOR
        FOR RESULT SET :PROJ_LOC;

EXEC SQL FETCH NEXT FROM PROJ_CSR FOR
          10 ROWS INTO :RS_Array;
 ...

EXEC SQL CLOSE PROJ_CSR;
```

- Result Set Procedure

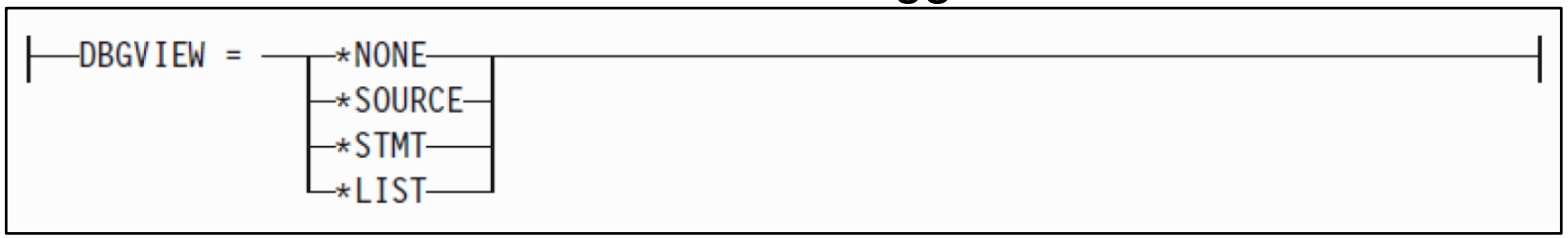
```
CREATE PROCEDURE GetProjects
 (iDept CHAR(3))
  DYNAMIC RESULT SETS 1
BEGIN
 DECLARE project_search CURSOR
  WITH RETURN FOR
  SELECT Projname, Prstaff
    FROM project
    WHERE deptno = iDept
    ORDER BY Projname;

 OPEN project_search ;
END;
```

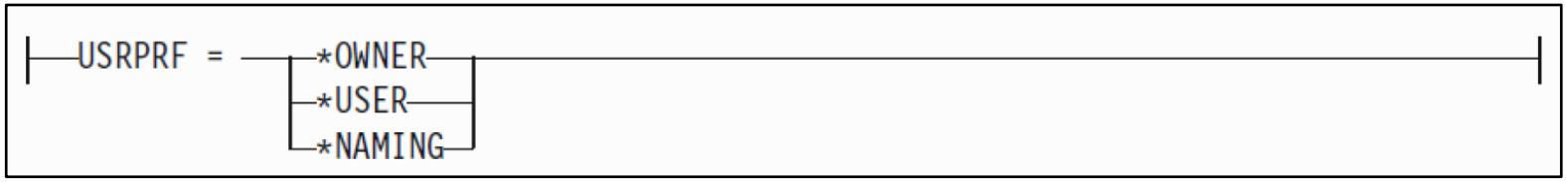
PROJNAME	PRSTAFF
W L PROD CONT PROGS	3.00
W L PROGRAM DESIGN	2.00
W L PROGRAMMING	9.00
W L ROBOT DESIGN	3.00

SET OPTION

- The SET OPTION statement can be used to set the internal precompiler options, much like the options on CRTSQLCI or CRTSLQRPGI
  - DBGVIEW = *SOURCE creates a debuggable version



  - USRPRF configures whether the user's or owner's authority is used (*OWNER) is default



PIPE statement

- PIPE is an alternate return mechanism for table function

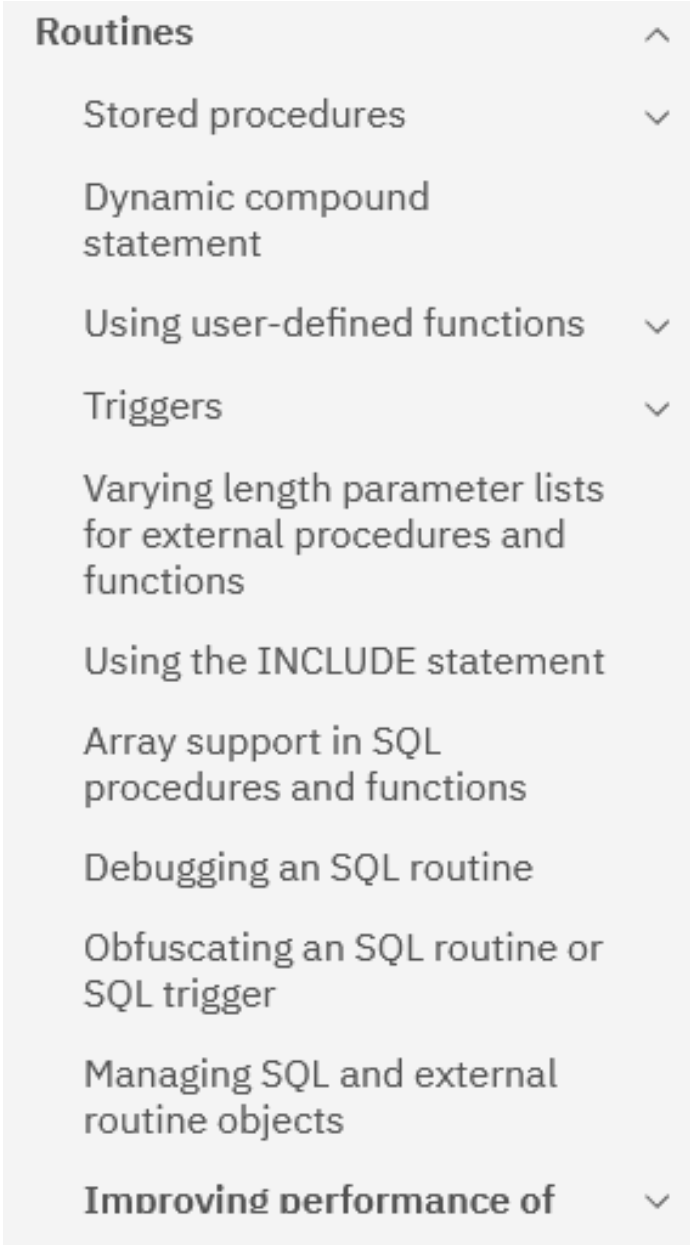
```
BEGIN
    FOR JRN_CURSOR CURSOR FOR SELECT X.*
                          FROM QSYS2.JOURNAL_RECEIVER_INFO X
                          WHERE DETACH_TIMESTAMP < CURRENT TIMESTAMP - DAYS_OLD DAYS
                              AND JOURNAL_RECEIVER_LIBRARY = RECEIVER_LIBRARY
                          ORDER BY DETACH_TIMESTAMP ASC, JOURNAL_RECEIVER_NAME ASC
    DO
      IF PREVIEW = 'NO' THEN
         PIPE ('YES', JOURNAL_RECEIVER_LIBRARY, JOURNAL_RECEIVER_NAME, DESCRIPTIVE_TEXT, JOURNAL_LIBRARY, JOURNAL_NAME,
      ELSE
         PIPE ('NO', JOURNAL_RECEIVER_LIBRARY, JOURNAL_RECEIVER_NAME, DESCRIPTIVE_TEXT, JOURNAL_LIBRARY, JOURNAL_NAME, S
      END IF;
    END FOR;

    RETURN;
END;
```

SQL Programming

- Concepts further explained
- Many examples of using SQL
- Detailed section on routines

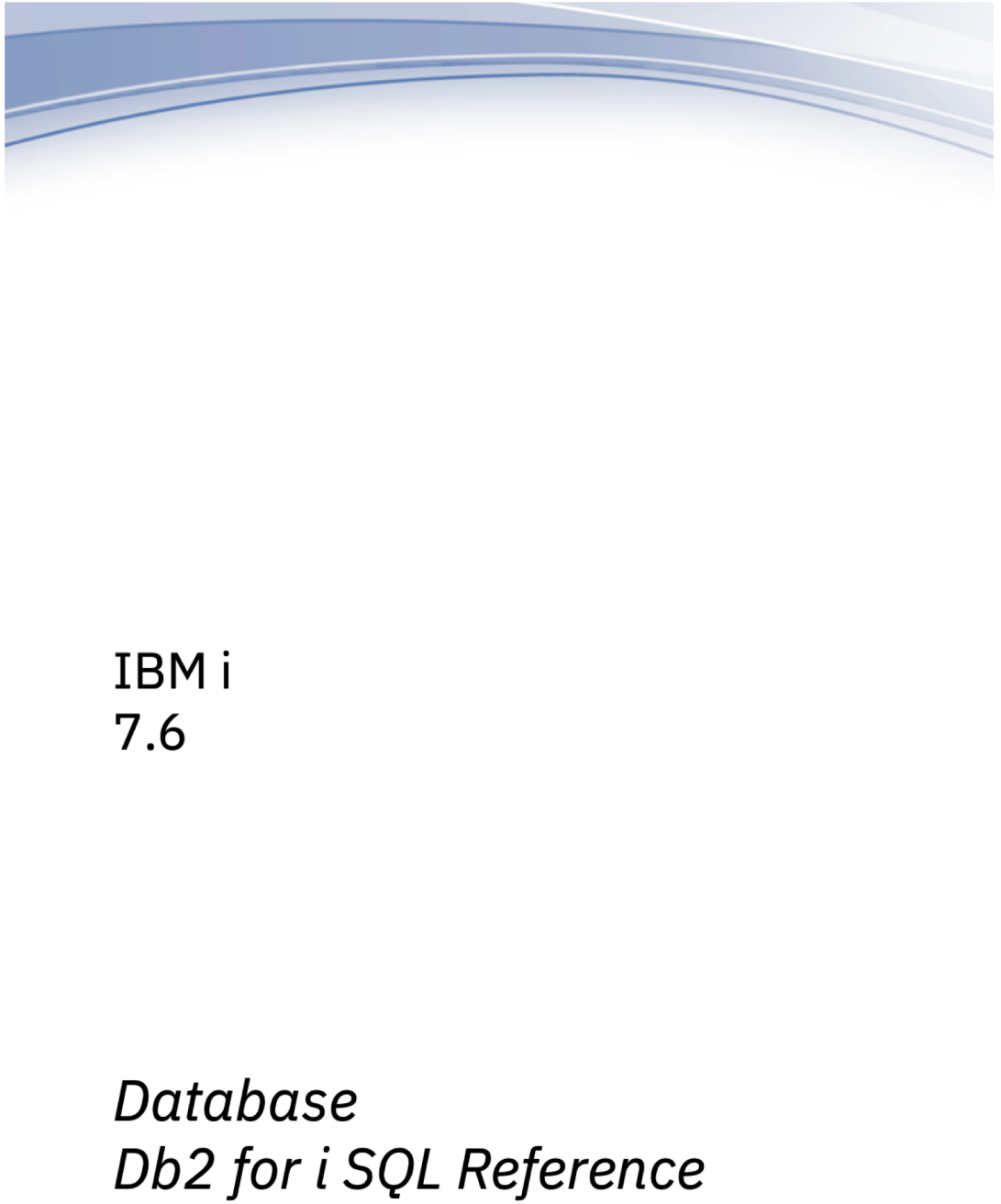
https://www.ibm.com/docs/en/i/7.6.0



SQL Reference

- 2,100 pages
- Updated twice each year

- Chapter 8 describes the SQL PL programming constructs



https://www.ibm.com/docs/en/ssw_ibm_i_76/pdf/rbafzpdf.pdf

Further Reading

Mad Scientist DB2 for i SQL :

- https://gist.github.com/NielsLiisberg

SQL Conductor Maestro:

- https://github.com/BirgittaHauser/Write-to-IFS-with-SQL
- https://github.com/BirgittaHauser/Generate-XML-and-JSON

Great SQL Table Function and Stored Proc Usage Examples :

- https://gist.github.com/forstie

The best place where I learned from :

- http://jplamontre.free.fr/AS400/UDTF.htm
- https://www.scottklement.com/udtf/

Good to know :

- https://www.ibm.com/docs/en/i/7.4.0?topic=applications-determining-equivalent-sql-ile-rpg-data-types
- https://www.rpgpgm.com/2018/09/sql-type-variables-in-rpg.html

Demo Time

Last Parting Thoughts...

“A thousand candles can be lighted from the flame of one candle, and the life of the candle will not be shortened. Happiness can be spread without diminishing that of yourself.”

— **Buddha or Ghandi (depending on Search Index on Search Engine if its up to date with its own AI processing)**

My Gratitude for Attending this session.

- Merci
- Danke
- Grazie
- Gracias
- Obrigado
- Dankie

to you all ☺

And final big thanks to
Ryan Moeller (IBM) & COMMON Europe

common

And a bonus (if you got here) :

https://tidycal.com/mlr-consulting/60-minute-meeting-m5rw2vn-1k62nqd-ibmi