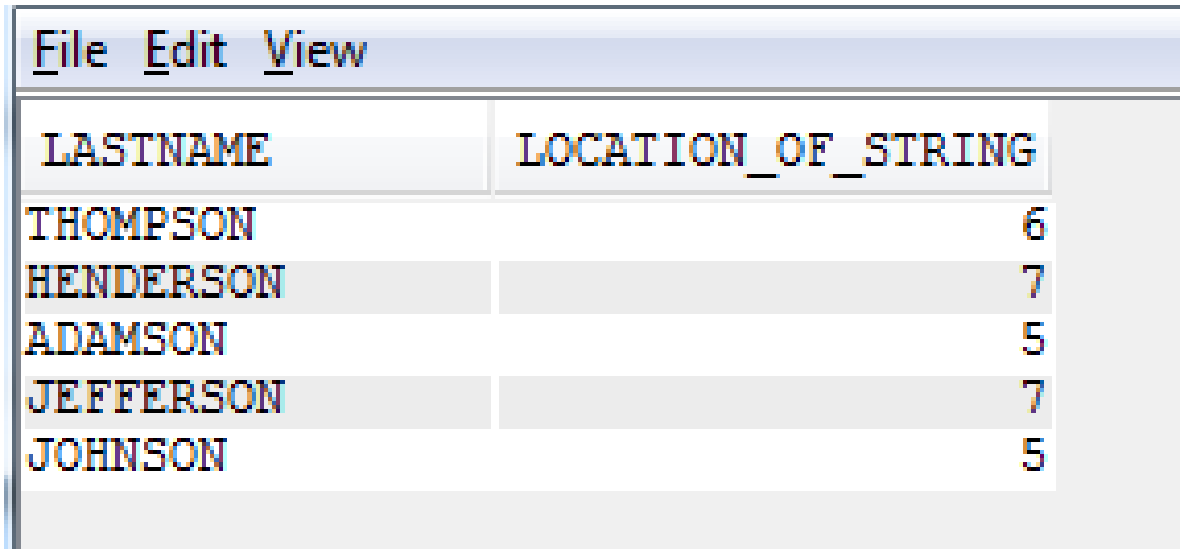
```
SELECT        lastName
             ,LOCATE_IN_STRING(lastName,   'SON', 1, 1)
              AS     Location_of_String
FROM          derrja.employee
WHERE          lastName LIKE('%SON%');
```



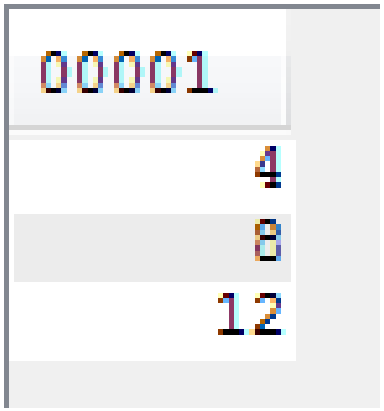
File Edit View

LASTNAME	LOCATION_OF_STRING
THOMPSON	6
HENDERSON	7
ADAMSON	5
JEFFERSON	7
JOHNSON	5

```
-- Find the location of the word AMERICAN in the string, and extract that word.
SELECT        lastName
            ,LOCATE_IN_STRING(lastName,   'SON', 1 , 1)         AS Starting_Location
            ,SUBSTR(lastName,  LOCATE_IN_STRING(lastName,  'SON', 1 , 1)
,LENGTH('SON')) AS extracted_value
FROM          derrja.employee
WHERE         LOCATE_IN_STRING(lastName,   'SON', 1, 1)   <> 0;
```

LASTNAME	STARTING_LOCATION	EXTRACTED_VALUE
THOMPSON	6	SON
HENDERSON	7	SON
ADAMSON	5	SON
JEFFERSON	7	SON
JOHNSON	5	SON

```
values locate_in_string('123.456.789.101', '.', 1, 1),
       locate_in_string('123.456.789.101', '.', 1, 2),
       locate_in_string('123.456.789.101', '.', 1, 3);
```



00001
4
8
12

```
-- LOCATE_IN_STRING (GLOBAL VARIABLE)

CREATE OR REPLACE VARIABLE derrja.gvString VARCHAR(100) DEFAULT 'SON';



-- Find the location of the word AMERICAN in the string, and extract that word.
SELECT        lastName
           ,LOCATE_IN_STRING(lastName,   derrja.gvString, 1 , 1)                AS Starting_Location
           ,SUBSTR(lastName,  LOCATE_IN_STRING(lastName,  derrja.gvString, 1 , 1) ,LENGTH(derrja.gvString)) AS
extracted_value
FROM          derrja.employee
WHERE         LOCATE_IN_STRING(lastName,   derrja.gvString, 1, 1)   <> 0;



SET  derrja.gvString = 'ER';
```

LASTNAME	STARTING_LOCATION	EXTRACTED_VALUE
THOMPSON	6	SON
HENDERSON	7	SON
ADAMSON	5	SON
JEFFERSON	7	SON
JOHNSON	5	SON

LASTNAME	STARTING_LOCATION	EXTRACTED_VALUE
GEYER	4	ER
STERN	3	ER
HENDERSON	5	ER
SPENSER	6	ER
WALKER	5	ER
JEFFERSON	5	ER
PEREZ	2	ER
SCHNEIDER	8	ER
PARKER	5	ER
HEMMINGER	8	ER
MONTEVERDE	7	ER
SPRINGER	7	ER

(MEMBER LIST)

```
SELECT        *
FROM          qsys2.SYSPARTITIONSTAT
WHERE         TABLE_NAME = 'QSOURCE'
  AND         TABLE_SCHEMA = 'DERRJA';
```

TABLE_SCHEMA	TABLE_NAME	TABLE_PARTITION	PARTITION_TYPE	PARTITION_NUMBER	PARTITION_TEXT
DERRJA	QSOURCE	DYNFTP		1	Dynamic FTP
DERRJA	QSOURCE	MSGSFL		2	-
DERRJA	QSOURCE	TSTJULIAN		3	Test JHA JHDATI cvt cal to jul da
DERRJA	QSOURCE	TSTSPECIAL		4	Test "SPECIAL" - same as virtual
DERRJA	QSOURCE	CRTUSRSPC		5	Create user space-not used-alread
DERRJA	QSOURCE	CRTUSRSPX		6	Create user space
DERRJA	QSOURCE	TESTUSRSPC		7	Create a user space and get the p
DERRJA	QSOURCE	RPGTMPL		8	-
DERRJA	QSOURCE	DSPFTMPL		9	-
DERRJA	QSOURCE	MPLFILE		10	-
DERRJA	QSOURCE	GOUT		11	-
DERRJA	QSOURCE	GOUT2		12	-
DERRJA	QSOURCE	TESTPRINTP		13	test print functions
DERRJA	QSOURCE	TESTPRINTR		14	test print functions
DERRJA	QSOURCE	TSTPRINTC		15	-

-ANALYSE ALL SQL STATEMENTS IN A PROGRAM LIBRARY TO FIND ALL PLACES WHERE A SPECIFIC FIELD IS UPDATED VIA SQL

STEP A: EXPLAIN “PARSE”
STEP B: FIND EVERY SQL STATEMENT IN EVERY PROGRAM
STEP C: PARSE THE DESIRED DATA (for instance, find every

STEP A: EXPLAIN “PARSE”

```
SELECT        *
FROM          TABLE(qsys2.parse_statement('SELECT     L.cfcif#, L.cfatyp, L.cfacc#,
                    L.cfsnme, R.*
FROM     datedw.cfacct L, datedw.ddmast R
WHERE    L.cfcif# = R.cifno
    AND   L.cfcif# <> ''x''',  '*SYS', '*PERIOD', '*APOSTSQL')) c;
```

NAME_TYPE	NAME	SCHEMA	RDB	COLUMN_NAME	USAGE_TYPE	NAME_START_POSITION	SQL_STATEMENT_TYPE
COLUMN	CFACCT	DATEDW	-	CFCIF#	QUERY	14	QUERY
COLUMN	CFACCT	DATEDW	-	CFATYP	QUERY	24	QUERY
COLUMN	CFACCT	DATEDW	-	CFACC#	QUERY	34	QUERY
COLUMN	CFACCT	DATEDW	-	CFSNME	QUERY	44	QUERY
TABLE	CFACCT	DATEDW	-	-	QUERY	113	QUERY
TABLE	DDMAST	DATEDW	-	-	QUERY	187	QUERY
COLUMN	CFACCT	DATEDW	-	CFCIF#	QUERY	262	QUERY
COLUMN	DDMAST	DATEDW	-	CIFNO	QUERY	273	QUERY
COLUMN	CFACCT	DATEDW	-	CFCIF#	QUERY	337	QUERY

STEP B: FIND EVERY SQL STATEMENT IN EVERY PROGRAM

```
WITH T1 AS (
 SELECT         Program_Schema, Program_Name, Program_Type, Statement_Text
FROM           QSYS2.SYSPROGRAMSTMTSTAT
WHERE           PROGRAM_SCHEMA IN ('RUNCUS','RUNCUSEDW','DERRJA','MUICKA')
  AND          STATEMENT_TEXT <> '‘)

--STEP C: PARSE THE DESIRED DATA (for instance, find every updating statement

SELECT         T1.Program_Schema, T1.Program_Name, T1.Program_Type,
T1.Statement_Text, c.Name, c.Schema, c.Column_Name, SQL_Statement_Type
FROM           T1,
              TABLE(qsys2.parse_statement(Statement_Text, '*SYS', '*PERIOD',
'*APOSTSQL')) c
WHERE           c.SQL_Statement_Type IN ('INSERT','UPDATE')
  AND          c.Name         IN ('DDMAST', 'LNMAST')
  AND          c.Column_Name   = 'CBAL'
ORDER BY       c.Column_Name;
```

PROGRAM_SCHEMA	PROGRAM_NAME	PROGRAM_TYPE	STATEMENT_TEXT
ERRJA	BE0547TEST	*PGM	SELECT CAST (MIN (CASE WHEN RENDT > 0 THEN RENDT WHEN ORGDT > 0 THEN ORGDT E...
ERRJA	BE0589	*PGM	DECLARE CSR_1 CURSOR FOR SELECT CIFNO , ACCTNO , ACTYPE , ADDNAM , ALTNAM , AL...
ERRJA	BE0589	*PGM	SET : H = TRIM (MONTHNAME (CURDATE ( ))) || ' ' || DAY (CURDATE ( )) || ...
ERRJA	BE0589	*PGM	OPEN CSR_1
ERRJA	BE0589	*PGM	FETCH NEXT FROM CSR_1 FOR : H ROWS INTO : H
ERRJA	BE0589	*PGM	CLOSE CSR_1
ERRJA	BE0594	*PGM	DECLARE CSR_1 CURSOR FOR SELECT CIFNO , ACCTNO , ACTYPE , ADDNAM , ALTNAM , AL...
ERRJA	BE0594	*PGM	OPEN CSR_1
ERRJA	BE0594	*PGM	FETCH NEXT FROM CSR_1 FOR : H ROWS INTO : H
ERRJA	BE0594	*PGM	CLOSE CSR_1
ERRJA	BE0642TST	*PGM	DECLARE LNCURSOR CURSOR FOR SELECT M . * , COALESCE (H . LHEFD6 , 000000) FR...
ERRJA	BE0642TST	*PGM	OPEN LNCURSOR
ERRJA	BE0642TST	*PGM	FETCH NEXT FROM LNCURSOR INTO : H ,: H ,: H ,: H ,: H ,: H ,: H ,: H ,: H...
ERRJA	BE0642TST	*PGM	CLOSE LNCURSOR
ERRJA	BE0684T	*PGM	DECLARE USRLIST CURSOR FOR SELECT * FROM MUICKA . BE0677F WHERE ID = : H FOR U...
ERRJA	BE0684T	*PGM	SET TRANSACTION ISOLATION LEVEL NC

STEP B: FIND EVERY SQL STATEMENT IN EVERY PROGRAM

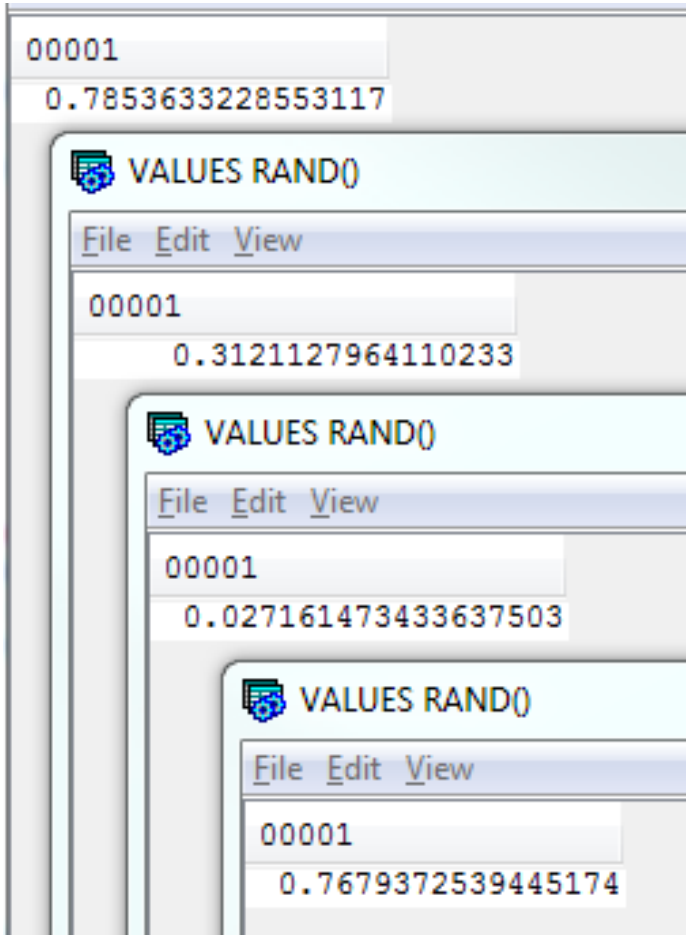
PROGRAM_SCHEMA	PROGRAM_NAME	PROGRAM_TYPE	STATEMENT_TEXT
DERRJA	BE0547TEST	*PGM	SELECT CAST (MIN (CASE WHEN RENDT > 0 THEN REN...
DERRJA	BE0589	*PGM	DECLARE CSR_1 CURSOR FOR SELECT CIFNO , ACCTNO ,...
DERRJA	BE0589	*PGM	SET : H = TRIM (MONTHNAME (CURDATE ( ))) || ...
DERRJA	BE0589	*PGM	OPEN CSR_1
DERRJA	BE0589	*PGM	FETCH NEXT FROM CSR_1 FOR : H ROWS INTO : H
DERRJA	BE0589	*PGM	CLOSE CSR_1
DERRJA	BE0594	*PGM	DECLARE CSR_1 CURSOR FOR SELECT CIFNO , ACCTNO ,...
DERRJA	BE0594	*PGM	OPEN CSR_1
DERRJA	BE0594	*PGM	FETCH NEXT FROM CSR_1 FOR : H ROWS INTO : H
DERRJA	BE0594	*PGM	CLOSE CSR_1
DERRJA	BE0642TST	*PGM	DECLARE LNCURSOR CURSOR FOR SELECT M . * , COALE...
DERRJA	BE0642TST	*PGM	OPEN LNCURSOR
DERRJA	BE0642TST	*PGM	FETCH NEXT FROM LNCURSOR INTO : H ,: H ,: H ,: H...
DERRJA	BE0642TST	*PGM	CLOSE LNCURSOR
DERRJA	BE0684T	*PGM	DECLARE USRLIST CURSOR FOR SELECT * FROM MUICKA ...
DERRJA	BE0684T	*PGM	SET TRANSACTION ISOLATION LEVEL NC
DERRJA	BE0684T	*PGM	VALUES (USER) INTO : H
DERRJA	BE0684T	*PGM	OPEN USRLIST
DERRJA	BE0684T	*PGM	FETCH USRLIST INTO : H ,: H ,: H ,: H ,: H ,: H ...
DERRJA	BE0684T	*PGM	DELETE ATMUSER WHERE USR = : H AND BANKNO = : H
DERRJA	BE0684T	*PGM	DELETE ISUSRINF WHERE USRNAM = : H AND BNKNBR = ...

STEP C: PARSE THE DESIRED DATA (for instance, find every updating statement

PROGRAM_SCHEMA	PROGRAM_NAME	PROGRAM_TYPE	STATEMENT_TEXT	NAME
RUNCUS	DD4300	*PGM	UPDATE DDMAST SET DDITEMSBAL = : H , DDNSFFEBAL ...	DDMAST
RUNCUS	DD4300	*PGM	UPDATE DDMAST SET DDITEMSBAL = : H , DDNSFFEBAL ...	DDMAST

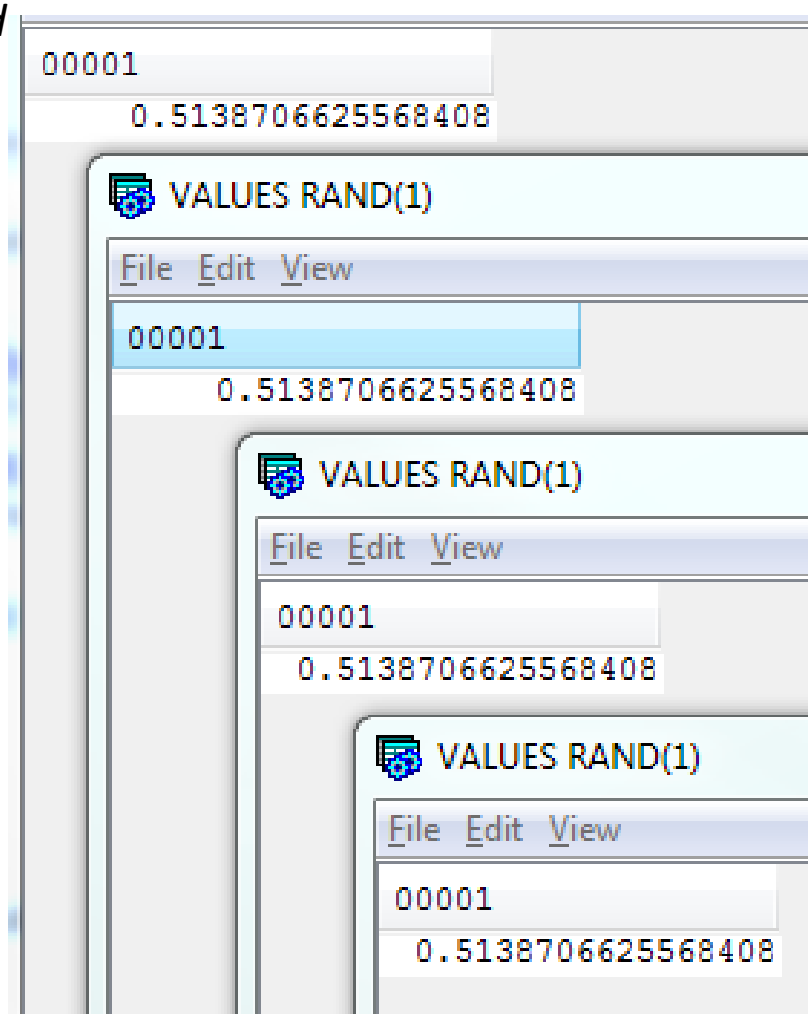
-- The RAND function returns a floating point value
-- greater than or equal to 0 and less than or equal to 1.
-- The following produces a different random number every time it is run.

```
VALUES RAND();
VALUES RAND();
VALUES RAND();
VALUES RAND();
VALUES RAND();
VALUES RAND();
```



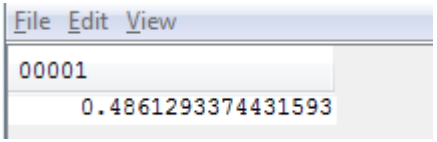
*-- A **specific seed value** will produce the same sequence of random numbers for a*
-- specific instance of a RAND function in a query each time the query is executed. If
-- a seed value is not specified, a different sequence of random numbers is produced
-- each time the query is executed

```
VALUES RAND(1);
VALUES RAND(1);
VALUES RAND(1);
VALUES RAND(1);
```



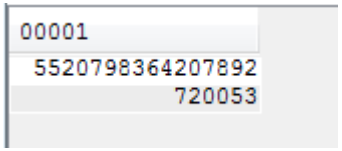
-- The value 2,147,483,647 seems to be the largest seed we can use, so we cannot use the account number as a seed value!

```
VALUES RAND(2147483647);
```



-- or as an integer

```
VALUES CAST((RAND(MICROSECOND(CURRENT_TIMESTAMP)) * 10000000000000000 )
AS BIGINT),    MICROSECOND(CURRENT_TIMESTAMP);
```



```
--EXAMPLE 1 INITIALLY DEFERRED
DROP TABLE   derrtest.empTemp;
DELETE FROM  derrtest.employee WHERE EmpNo = 300001;
create table derrtest.empTemp as (
   select     e.empno,
              e.firstnme,
              e.lastname,
              e.phoneno,
              d.deptno,
              substr(d.deptname, 1, 12) as department,
              d.mgrno
    from      derrtest.employee   e,
              derrtest.department d
    where     e.workdept = d.deptno)
DATA INITIALLY DEFERRED                    --<< "Refreshable-table-options"  See printed pg 1062
REFRESH DEFERRED                          --<< "Refreshable-table-options"
MAINTAINED BY USER;                        --<< "Refreshable-table-options"

  select * from derrtest.empTemp  ORDER BY empNo;;   -- Executed, nothing returned.
  refresh table derrtest.empTemp;                    -- Valid because the table is materialized.
  select * from derrtest.empTemp  ORDER BY empNo;;   -- Executed and 37 rows were returned.
```

```
--EXAMPLE 2 INITIALLY IMMEDIATE
DROP TABLE   derrtest.empTemp;
DELETE FROM  derrtest.employee WHERE EmpNo = 300001;
create table derrtest.empTemp as (
   select     e.empno,
              e.firstnme,
              e.lastname,
              e.phoneno,
              d.deptno,
              substr(d.deptname, 1, 12) as department,
              d.mgrno
    from      derrtest.employee   e,
              derrtest.department d
    where     e.workdept = d.deptno)
DATA INITIALLY IMMEDIATE                    --<< "Refreshable-table-options"  See printed pg 1062 (scanable page 1084)
REFRESH DEFERRED                          --<< "Refreshable-table-options"
MAINTAINED BY USER;                        --<< "Refreshable-table-options“

  select * from derrtest.empTemp  ORDER BY empNo;;          -- Executed and 37 rows were returned.
  refresh table derrtest.empTemp;                           -- Valid because the table is materialized.
  select * from derrtest.empTemp  ORDER BY empNo;;          -- Executed and 37 rows were returned.

-- Insert a row in the table that MQT is based.
INSERT INTO    derrtest.employee values(300001,'JOHN','A','HIRKO','D11',1234,'2017-05-11','CLERK',12,'M','1970-01-
01',25000.00,1000.00,2500.00);
SELECT * FROM  derrtest.empTemp ORDER BY empNo;   -- Added row is NOT THERE.
refresh table  derrtest.empTemp;
SELECT * FROM  derrtest.empTemp ORDER BY empNo;   -- Added row is returned
```

```
--EXAMPLE 3 MAINTAIN THE MQT DIRECTLY

UPDATE        derrtest.empTemp
SET           LastName = 'KWAN-CHANGED'
WHERE         empNo    = '000030';          -- executed ok

SELECT * from derrtest.employee ORDER BY empNo;   -- not updated in the base table.

INSERT INTO    derrtest.empTemp VALUES
          ('000001','JUAN','DERR','4321','B01','OPERATIONS','000090');      -- No error
SELECT * FROM  derrtest.empTemp ORDER BY empNo;   -- Add row is returned and KWAN
                                                     changed to KWAN-CHANGED
refresh table  derrtest.empTemp;
SELECT * FROM  derrtest.empTemp ORDER BY empNo;   -- '000001' is gone and name is
                                                      returned to KWAN.
```

```
--EXAMPLE 4 THE SUMMARY MQT

SELECT * FROM  derrtest.employee ORDER BY empNo;  -- Find a field to summarize
SELECT * FROM  derrtest.department ;
-- use field BONUS to group by

DROP TABLE   derrtest.empTemp2;
DELETE FROM  derrtest.employee WHERE EmpNo = 300001;
create table derrtest.empTemp2 as (
   select      e.workdept,
               substr(d.deptname, 1, 12) as department,
               SUM(e.Bonus) AS Ttl_Bonus,
               COUNT(*) AS #Empl_Dept
    from       derrtest.employee   e,
               derrtest.department d
    where      e.workdept = d.deptno
    GROUP BY   e.workdept,substr(d.deptname, 1, 12)
    ORDER BY   e.workdept,substr(d.deptname, 1, 12))
DATA INITIALLY IMMEDIATE                                --<< "Refreshable-table-options"
REFRESH DEFERRED                                        --<< "Refreshable-table-options"
MAINTAINED BY USER;                                     --<< "Refreshable-table-options"


 select * from derrtest.empTemp2;                       -- Executed and 7 rows were returned. D11 = 5500.00

-- Insert a row in the table that MQT is based.
INSERT INTO    derrtest.employee values(300001,'JOHN','A','HIRKO','D11',1234,'2017-05-11','CLERK',12,'M','1970-01-
01',25000.00,1000.00,2500.00);
SELECT * FROM  derrtest.empTemp2;                       -- 7 rows were returned. D11 = 5500.00
refresh table  derrtest.empTemp2;
SELECT * FROM  derrtest.empTemp2;                       -- 7 rows were returned. D11 = 6500.00
```

EXAMPLE 6 (A) Standard (simple) result set

```
CREATE OR REPLACE PROCEDURE derrja.spStandard(
   IN  pDepartment     VARCHAR(10))
   RESULT SET 1
   LANGUAGE SQL

BEGIN

 DECLARE    c1 CURSOR WITH RETURN TO CALLER FOR
  SELECT     empNo, firstNme, midInit, lastName, workDept, (SELECT       deptName
                                      FROM        department
                                      WHERE       deptNo  = pDepartment)
  FROM       employee
  WHERE      pDepartment = workDept;

 OPEN c1;
END;
```

(both return result sets, as some employees do not have an assigned department.)
CALL spStandard('D11');
CALL spStandard(''); (RETURNS NOTHING)

EMPNO	FIRSTNME	MIDINIT	LASTNAME	WORKDEPT	DEPTNAME
000060	IRVING	F	STERN	D11	MANUFACTURING SYSTEMS
000150	BRUCE		ADAMSON	D11	MANUFACTURING SYSTEMS
000160	ELIZABETH	R	PINKA	D11	MANUFACTURING SYSTEMS
000170	MASATOSHI	J	YOSHIMURA	D11	MANUFACTURING SYSTEMS
000180	MARILYN	S	SCOUTTEN	D11	MANUFACTURING SYSTEMS
000190	JAMES	H	WALKER	D11	MANUFACTURING SYSTEMS
000200	DAVID		BROWN	D11	MANUFACTURING SYSTEMS
000210	WILLIAM	T	JONES	D11	MANUFACTURING SYSTEMS
000220	JENNIFER	K	LUTZ	D11	MANUFACTURING SYSTEMS
200170	KIYOSHI		YAMAMOTO	D11	MANUFACTURING SYSTEMS
200220	REBA	K	JOHN	D11	MANUFACTURING SYSTEMS

EXAMPLE 6 (B) Proprietary result set

This appears to be the same as standard (simple) result set,
Except it sets the result set as a cursor.

```
CREATE OR REPLACE PROCEDURE derrja.spProprietary(
   IN  pDepartment    VARCHAR(10))
   RESULT SET 1
   LANGUAGE SQL

BEGIN

 DECLARE   c1 CURSOR FOR
 SELECT    empNo, firstNme, midInit, lastName, workDept, (SELECT      deptName
                                    FROM       department
                                    WHERE      deptNo  = pDepartment)
 FROM      employee
 WHERE     pDepartment = workDept;

 OPEN c1;

 SET RESULT SETS CURSOR c1;     -- I get the apparent exact same results whether or not this line is here!

END;

CALL spProprietary('D11');
CALL spProprietary('');
```

EMPNO	FIRSTNME	MIDINIT	LASTNAME	WORKDEPT	DEPTNAME
000060	IRVING	F	STERN	D11	MANUFACTURING SYSTEMS
000150	BRUCE		ADA...	D11	MANUFACTURING SYSTEMS
000160	ELIZABETH	R	PINKA	D11	MANUFACTURING SYSTEMS
000170	MASATOSHI	J	YO...	D11	MANUFACTURING SYSTEMS
000180	MARILYN	S	SCO...	D11	MANUFACTURING SYSTEMS
000190	JAMES	H	WALKER	D11	MANUFACTURING SYSTEMS
000200	DAVID		BROWN	D11	MANUFACTURING SYSTEMS
000210	WILLIAM	T	JONES	D11	MANUFACTURING SYSTEMS
000220	JENNIFER	K	LUTZ	D11	MANUFACTURING SYSTEMS
200170	KIYOSHI		YAM...	D11	MANUFACTURING SYSTEMS
200220	REBA	K	JOHN	D11	MANUFACTURING SYSTEMS

EXAMPLE 6 (C) LOCATOR

```
/* This procedure processes the result set from another stored procedure, using the
   RESULT_SET_LOCATOR declaration and the ASSOCIATE RESULT SET LOCATORS.

   It reads the result set from a stored procedure into a cursor in a separate procedure.
*/

DROP TABLE qtemp.tstLocator;

CREATE TABLE QTemp.tstLocator (
    empNo       VARCHAR(10),
    firstNme    VARCHAR(50),
    midInit     VARCHAR(10),
    lastName    VARCHAR(50),
    workDept    VARCHAR(3),
    deptName    VARCHAR(50));
```

```
CREATE OR REPLACE PROCEDURE derrja.spLocator(
   IN  pDepartment    VARCHAR(10))
   RESULT SET 1
   LANGUAGE SQL
 ------------------------------------------------------------------------------------------------
BEGIN
 DECLARE   sprs1      RESULT_SET_LOCATOR VARYING;
 DECLARE   empNo      VARCHAR(10);
 DECLARE   firstNme   VARCHAR(50);
 DECLARE   midInit    VARCHAR(10);
 DECLARE   lastName   VARCHAR(50);
 DECLARE   workDept   VARCHAR(3);
 DECLARE   deptName   VARCHAR(50);
 DECLARE   row_not_found INT;
---------------------------------------- ERROR HANDLERS ---------------------------------------
-- Test for EOF
 DECLARE CONTINUE HANDLER FOR SQLSTATE '02000'
    SET row_not_found = 1;
---------------------------------------- PROCESSING -------------------------------------------
 CALL spProprietary(pDepartment);
 ASSOCIATE RESULT SET LOCATORS(sprs1) WITH PROCEDURE derrja.spProprietary;
 ALLOCATE mycur CURSOR FOR RESULT SET sprs1;

 myloop:
 LOOP
  FETCH mycur INTO empNo, firstNme, midInit, lastName, workDept, deptName;

  IF row_not_found = 1 THEN
   LEAVE myloop;
  END IF;

  -- DO SOMETHING WITH THE DATA YOU JUST READ  --
  -- IN THIS EXAMPLE, IN ORDER TO PROVE THAT I ACTUALLY RETRIEVED DATA. I SIMPLY WRITE TO QTEMP --
  INSERT INTO qtemp.tstLocator VALUES(empNo, firstNme, midInit, lastName, workDept, deptName);

 END LOOP;
 CLOSE myCur;

END;
```

```
CALL spLocator('D11');                    -- simply adds to the qtemp table
CALL spLocator('B01');                    -- simply adds to the qtemp table
SELECT * FROM qtemp.tstLocator;
```

```
-- DATE STUFF (TIMESTAMP_FORMAT)
-- Date Stuff using new Timestamp_Format scalar function

-- NOTICE THE USE OF RR INSTEAD OF YY WHEN USING A TWO POSITION YEAR.
--  ALWAYS USE RR INSTEAD OF YY WHEN YOU HAVE A 2 DIGIT YEAR!
--  YY WILL CONVERT THE DATE INCORRECTLY:
--      USING YY 080789       converts to 2089-08-07
--      USING RR 080789 converts to 1989-08-07


CREATE OR REPLACE TABLE  derrja.datetest  (
 mmddyy   DECIMAL(6, 0)
,ddmmyy   DECIMAL(6, 0)
,yymmdd   DECIMAL(6, 0)
,mmddyyyy DECIMAL(8, 0)
,ddmmyyyy DECIMAL(8, 0)
,yyyymmdd DECIMAL(8, 0)
,yyddd    DECIMAL(5, 0)
,yyyyddd  DECIMAL(7, 0));

TRUNCATE derrja.datetest;

--                        mmddyy ddmmyy yymmdd  MMDDYYYY DDMMYYYY YYYYMMDD YYDDD YYYYDDD
INSERT INTO derrja.datetest VALUES(110513, 051113, 131105, 11052013, 05112013, 20131105, 13309, 2013309)
                          ,(110593, 051193, 931105, 11051993, 05111993, 19931105, 93309, 1993309)
                          ,(101690, 161090, 901016, 10161990, 15101990, 19901016, 90289, 1990289)
                          ,(080789, 070889, 890807, 08071989, 07081989, 19890807, 89219, 1989219)
                          ,(022037, 200237, 370220, 02201937, 20021937, 19370220, 37051, 1937051) ;
```

MMDDYY	DDMMYY	YYMMDD	MMDDYYYY	DDMMYYYY	YYYYMMDD	YYDDD	YYYYDDD
110513	51113	131105	11052013	5112013	20131105	13309	2013309
110593	51193	931105	11051993	5111993	19931105	93309	1993309
101690	161090	901016	10161990	15101990	19901016	90289	1990289
80789	70889	890807	8071989	7081989	19890807	89219	1989219
22037	200237	370220	2201937	20021937	19370220	37051	1937051

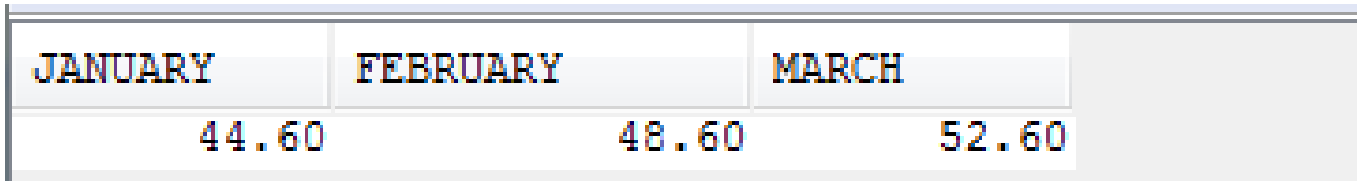
```
SELECT
     mmddyy, CASE WHEN SUBSTR(DIGITS(mmddyy), 5, 2) < 40
          THEN Date(Timestamp_Format(DIGITS(mmddyy), 'MMDDYY'))

          ELSE Date(Timestamp_Format( SUBSTR(DIGITS(mmddyy)  ,1,4) ||
                    '19' ||
                    SUBSTR(DIGITS(mmddyy),5,2)  , 'MMDDYYYY'))
          END AS MMDDYY_Out
    ,ddmmyy,   Date(Timestamp_Format(DIGITS(ddmmyy),   'DDMMRR'))   AS DDMMYY_Out
    ,yymmdd,   Date(Timestamp_Format(DIGITS(yymmdd),   'RRMMDD'))   AS YYMMDD_Out
    ,mmddyyyy, Date(Timestamp_Format(DIGITS(MMDDYYYY), 'MMDDYYYY')) AS MMDDYYYY_Out
    ,ddmmyyyy, Date(Timestamp_Format(DIGITS(DDMMYYYY), 'DDMMYYYY')) AS DDMMYYYY_Out
    ,yyyymmdd, Date(Timestamp_Format(DIGITS(YYYYMMDD), 'YYYYMMDD')) AS YYYYMMDD_Out
    ,yyyyddd,  DATE(cast(YYYYDDD as char(7)))  AS YYYYDDD_Out
from  derrja.datetest;
```

MMDDYY	MMDDYY_OUT	DDMMYY	DDMMYY_OUT	YYMMDD	YYMMDD_OUT	MMDDYYYY	MMDDYYYY_OUT	DDMMYYYY	DDMMYYYY_OUT	YYYYMMDD	YYYYMMDD_OUT	YYYYDDD	YYYYDDD_OUT
110513	2013-11-05	51113	2013-11-05	131105	2013-11-05	11052013	2013-11-05	5112013	2013-11-05	20131105	2013-11-05	2013309	2013-11-05
110593	1993-11-05	51193	1993-11-05	931105	1993-11-05	11051993	1993-11-05	5111993	1993-11-05	19931105	1993-11-05	1993309	1993-11-05
101690	1990-10-16	161090	1990-10-16	901016	1990-10-16	10161990	1990-10-16	15101990	1990-10-15	19901016	1990-10-16	1990289	1990-10-16
80789	1989-08-07	70889	1989-08-07	890807	1989-08-07	8071989	1989-08-07	7081989	1989-08-07	19890807	1989-08-07	1989219	1989-08-07
22037	2037-02-20	200237	2037-02-20	370220	2037-02-20	2201937	1937-02-20	20021937	1937-02-20	19370220	1937-02-20	1937051	1937-02-20

PIVOTING THE DATA - SIMPLE EXAMPLE

```
drop table derrtest.TABLEA;

CREATE OR REPLACE TABLE derrtest.TABLEA (
   Code_Month   INT,
   Code_Sales   DECIMAL(8 , 2) );

TRUNCATE TABLE derrtest.TABLEA;


INSERT INTO  derrtest.TABLEA VALUES
   (01,  11.00),
   (02,  12.00),
   (03,  13.00),
   (01,  11.10),
   (02,  12.10),
   (03,  13.10),
   (01,  11.20),
   (02,  12.20),
   (03,  13.20),
   (01,  11.30),
   (02,  12.30),
   (03,  13.30);
```

```
SELECT * FROM derrtest.TABLEA;

WITH
T1 AS (
SELECT        Code_Month, SUM(Code_Sales) AS Tot_Sales
FROM          derrtest.TABLEA
GROUP BY CUBE   (Code_Month))                          -- SELECT * FROM T1;                -- show this
first
SELECT        COALESCE(MAX(CASE    WHEN    Code_Month = 01 THEN Tot_Sales END), 0) AS January,
              COALESCE(MAX(CASE    WHEN    Code_Month = 02 THEN Tot_Sales END), 0) AS February,
              COALESCE(MAX(CASE    WHEN    Code_Month = 03 THEN Tot_Sales END), 0) AS March
from          T1;
```



JANUARY	FEBRUARY	MARCH
44.60	48.60	52.60

REGEXP_COUNT

```
-- The REGEXP_COUNT function returns a count of the number of times that a
-- regular expression pattern is matched in a string.

-- Count the number of times "Steven" or "Stephen" occurs in the string "Steven
-- Jones and Stephen Smith are the best players".

VALUES REGEXP_COUNT('Steven Jones and Stephen Smith are the best players','Ste(v|ph)en');

-- Find those with duplicates of specific letters
SELECT * FROM DERRJA.EMPLOYEE;
SELECT empno, LastName, REGEXP_COUNT(LastName, 'CC|NN|FF|MM') FROM DERRJA.EMPLOYEE;

-- Set up the test
UPDATE DERRJA.EMPTEST SET LASTNAME = 'HAAAS' WHERE EmpNo = '000010';
UPDATE DERRJA.EMPTEST SET LASTNAME = 'LUUCCHESSII' WHERE EmpNo = '000110';

-- Find the number of ANY duplicate characters.
SELECT LastName, REGEXP_COUNT(LastName, '(.)\1+') FROM DERRJA.EMPTEST;              -- YES
SELECT LastName, REGEXP_COUNT(LastName, '(\w)\1+') FROM DERRJA.EMPTEST;              -- YES


SELECT  Lastname,
      CASE REGEXP_COUNT(LastName, '(\w)\1+')  WHEN 1  THEN 'Single duplicate character'
                                 WHEN 2  THEN 'Two duplicate characters'
                                 WHEN 3  THEN 'Three duplicate characters'
                                 ELSE       'Many duplicate characters'
                                 END
FROM DERRJA.EMPTEST WHERE REGEXP_COUNT(LastName, '(\w)\1+') > 0;
```

-- *The REGEXP_INSTR returns the starting position or the position after the end of*
-- *the matched substring, depending on the value of the return_option argument.*

-- REGEXP_INSTR(source-string, pattern-expression, start, occurence, return-option, flags,group)

-- **return-option**
-- An expression that specifies whether to return the starting position or the position after the end of the string that matches the pattern.
-- The expression must return a value of any built-in numeric, character-string, or graphic-string data type. The argument is cast to INTEGER
-- before evaluating the function. For more information about converting to INTEGER, see INTEGER or INT. The value of the integer must
-- be equal to 0 or 1. A value of 0 returns the starting position of the occurrence. A value of 1 returns the ending position of the occurrence.
-- If *return-option* is not specified, the default value is 0.

-- Find the position of the first instance of a character that precedes the letter o.
-- 1
-- ....0....
VALUES REGEXP_INSTR('hello to you', '.o',1,1); -- answer = 4 (found in the fourth position)

-- Find the position of the second instance of a character that preceeds the letter o.
VALUES REGEXP_INSTR('hello to you', '.o',1,2); -- answer = 7 (found in the seventh position)

-- Find the position of the third instance of a character that preceeds the letter o.
VALUES REGEXP_INSTR('hello to you', '.o',1,3); -- answer = 10 (found in the tenth position)

-- Find the position after the third occurrence of the first capture group
-- of the regular expression '(.o).' using case insensitive matching.
VALUES REGEXP_INSTR('hello to you', '(.o).', 1,3,1,'i',1);

CONVERT IP ADDRESS
--Integer 213739506

--How to convert
--To convert an IP address to integer, break it into four octets. For example, the ip address you provided can be broken into

--First Octet: 12
--Second Octet: 189
--Third Octet: 103
--Fourth Octet: 242

--To calculate the decimal address from a dotted string, perform the following calculation.

-- (first octet * 256^3) + (second octet * 256^2) + (third octet * 256) + (fourth octet)
-- = (first octet * 16777216) + (second octet * 65536) + (third octet * 256) + (fourth octet)
-- = (12 * 16777216) + (189 * 65536) + (103 * 256) + (242)
-- = 213739506

--
--12.189.103.242

```
--INTERACTIVE
CREATE OR REPLACE VARIABLE derrja.IPAddress VARCHAR(20) DEFAULT '12.189.103.242';
SET derrja.IPAddress = '10.0.0.3';

WITH T1 AS (
   SELECT
         CAST(SUBSTR(derrja.IPAddress,
            1,
            REGEXP_INSTR(derrja.IPAddress, '\.', 1, 1) -1) AS BIGINT) * (256 * 256 * 256)
       , CAST(SUBSTR(derrja.IPAddress,
            REGEXP_INSTR(derrja.IPAddress, '\.', 1, 1) + 1,
            REGEXP_INSTR(derrja.IPAddress, '\.', 1, 2) - (REGEXP_INSTR(derrja.IPAddress, '\.', 1, 1) +1)) AS BIGINT) *
(256 * 256)
       , CAST(SUBSTR(derrja.IPAddress,
            REGEXP_INSTR(derrja.IPAddress, '\.', 1, 2) + 1,
            REGEXP_INSTR(derrja.IPAddress, '\.', 1, 3) - (REGEXP_INSTR(derrja.IPAddress, '\.', 1, 2) +1)) AS BIGINT) *
256
       , CAST(SUBSTR(derrja.IPAddress,
            REGEXP_INSTR(derrja.IPAddress, '\.', 1, 3) + 1) AS BIGINT) AS LongIP
FROM       sysibm.sysdummy1),
T2 AS (
SELECT     (SELECT MAX(ip) FROM derrtest.ip2Nation WHERE ip <= LongIP) AS LongIP
FROM       T1)
SELECT     COALESCE(Cntry.Country, 'Not Found')  --LongIP, Code, ISO_Code_2, ISO_Code_3, ISO_Country,   ,
Cntry.Lat, Cntry.Lon
FROM       T2 LEFT OUTER JOIN derrTest.ip2Nation Nat
  ON       LongIP = Nat.ip
         LEFT OUTER JOIN derrtest.ip2NationCountries Cntry
  ON       LongIP = Nat.ip AND Nat.Country = Cntry.Code;
```

```
--Function
CREATE OR REPLACE VARIABLE derrja.IPAddress VARCHAR(20) DEFAULT '12.189.103.242';
-------- FUNCTION CALL
CREATE OR REPLACE FUNCTION derrja.fnGetCountry (
   IPAddress   VARCHAR(20))
   RETURNS     VARCHAR(256)
   LANGUAGE    SQL
   SPECIFIC    derrja.fnGetCountry
   NOT DETERMINISTIC
   SET OPTION
         DBGVIEW = *SOURCE
   START:
BEGIN
RETURN
WITH T1 AS (
   SELECT
         CAST(SUBSTR(IPAddress,
            1,
            REGEXP_INSTR(IPAddress, '\.', 1, 1) -1) AS BIGINT) * (256 * 256 * 256)
       + CAST(SUBSTR(derrja.IPAddress,
            REGEXP_INSTR(IPAddress, '\.', 1, 1) + 1,
            REGEXP_INSTR(IPAddress, '\.', 1, 2) - (REGEXP_INSTR(IPAddress, '\.', 1, 1) +1)) AS BIGINT) * (256 * 256)
       + CAST(SUBSTR(IPAddress,
            REGEXP_INSTR(IPAddress, '\.', 1, 2) + 1,
            REGEXP_INSTR(IPAddress, '\.', 1, 3) - (REGEXP_INSTR(IPAddress, '\.', 1, 2) +1)) AS BIGINT) * 256
       + CAST(SUBSTR(IPAddress,
            REGEXP_INSTR(IPAddress, '\.', 1, 3) + 1) AS BIGINT) AS LongIP
FROM       sysibm.sysdummy1)
SELECT     /*LongIP, Code, ISO_Code_2, ISO_Code_3, ISO_Country,*/ COALESCE(Cntry.Country, IPAddress || ' : ' || LongIP || ' Not Found')
/*, Cntry.Lat, Cntry.Lon*/
FROM       T1 LEFT OUTER JOIN derrTest.ip2Nation Nat
 ON        LongIP = Nat.ip
         LEFT OUTER JOIN derrtest.ip2NationCountries Cntry
 ON        LongIP = Nat.ip AND Nat.Country = Cntry.Code;
END;
```

values derrja.fnGetCountry('209.85.227.147');

values derrja.fnGetCountry('169.254.3.1');

values derrja.fnGetCountry('146.185.223.45');

values derrja.fnGetCountry('10.0.0.3');

select * from derrtest.ip2Nation where ip = 2466176813;