|  |  |
| --- | --- |
| **SQL query solution patterns**  -what can be done with LEFT/RIGHT OUTER JOIN can also be done by a correlated scalar subquery in the select list that can then be renamed with an alias. But in-efficient. You repeat the scalar subquery for every value to be fetched.  -CROSS JOIN used for generating sequence of values (either using ROW\_NUMBER or table of 0 to 9 values) with TOP for cutoff, mimicking a loop and transposing/pivoting/unpivoting. For some recursive problems, recursive CTE can be used in place of Loop.  CROSS JOIN/CROSS APPLY not only is used to generate Getnums and then CROSS APPLY can be used with GetNums to mimic a loop.  -when performing JOINs, think carefully before using AGGREGATE functions if the relationship is not one-to-one. Row repetition! Tables that form the ‘many’ part of the relationship have to be aggregated before joining.  -FULL OUTER JOIN is the UNION of results of LET and RIGHT OUTER JOINs.  -use EXISTS and NOT EXISTS where ever you IN and NOT IN being used. |  |

Examples:

|  |  |
| --- | --- |
| CROSS APPLY is deemed to be equal to CROSS JOIN as both do a Cartesian product and when the right hand side result set is empty, the corresponding row from left hand side is also removed from query result. But CROSS JOIN can use co-related subqueries which CROSS JOIN can’t.  SELECT \*  FROM (VALUES(1)) AS A(N)  CROSS APPLY (SELECT \* FROM (VALUES(1)) AS C(NN) WHERE C.NN = A.N) AS B(NN)  SELECT \*  FROM (VALUES(1)) AS A(N)  CROSS JOIN (SELECT \* FROM (VALUES(1)) AS C(NN) WHERE C.NN = A) AS B(NN) |  |
| **Loop examples using recursive CTE and CROSS APPLY**:  --USING CROSS APPLY WITH GETNUMS  DECLARE @STR AS VARCHAR(10) = 'KINGS';  SELECT SUBSTRING(@STR, S.N, 1) AS S  FROM (VALUES(1)) AS DUMMY(D)  CROSS APPLY (SELECT \* FROM GetNums(1, LEN(@STR))) AS S(N)  --USING RECURSIVE CTE  DECLARE @STARTPOS AS INT = 1;  ;WITH CTER  AS(  SELECT SUBSTRING(@STR, @STARTPOS, 1) AS S, @STARTPOS AS L  UNION ALL  SELECT SUBSTRING(@STR, L + 1 , 1) AS S, L + 1  FROM CTER AS R  WHERE SUBSTRING(@STR, L + 1 , 1) <> ''  )  SELECT \*  FROM CTER  SELECT TOP(500000) \*, CAST(CONCAT(A.A,B.B,C.C,D.D,E.E,F.F) AS INT) + 1 AS NUM  FROM (VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9)) AS A(A)  CROSS JOIN (SELECT \* FROM (VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9)) AS CP(A)) AS B(B)  CROSS JOIN (SELECT \* FROM (VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9)) AS CP(A)) AS C(C)  CROSS JOIN (SELECT \* FROM (VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9)) AS CP(A)) AS D(D)  CROSS JOIN (SELECT \* FROM (VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9)) AS CP(A)) AS E(E)  CROSS JOIN (SELECT \* FROM (VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9)) AS CP(A)) AS F(F)  ORDER BY NUM  WITH  L0 AS (SELECT 0 AS C FROM (VALUES(1),(1)) AS D(C)),--2  L1 AS (SELECT 0 AS C FROM L0 AS A CROSS JOIN L0 AS B ),--4  L2 AS (SELECT 0 AS C FROM L1 AS A CROSS JOIN L1 AS B ),--16  L3 AS (SELECT 0 AS C FROM L2 AS A CROSS JOIN L2 AS B ),--256  L4 AS (SELECT 0 AS C FROM L3 AS A CROSS JOIN L3 AS B ),--65536  L5 AS (SELECT 0 AS C FROM L4 AS A CROSS JOIN L4 AS B ),--4294967296  NUMS AS (SELECT ROW\_NUMBER() OVER(ORDER BY (SELECT NULL)) AS ROWNUM FROM L5)  SELECT TOP(500000 - 0 + 1) 0 + ROWNUM -1 AS N  FROM NUMS  ORDER BY ROWNUM | --CONCATENATE COLUMN VALUES TO A LIST USING RECURSIVE CTE  ;WITH CTE AS  (  SELECT \*  FROM (VALUES('LAPTOP', 'DELL'), ('LAPTOP', 'HP'), ('LAPTOP', 'COMPAQ'), ('MAC', 'APPLE')) AS P(PR, SUPP)  )  ,CTE1 AS  (  SELECT \*, ROW\_NUMBER() OVER(PARTITION BY PR ORDER BY SUPP) AS RN  FROM CTE  ),  RECURSIVECTE AS  (  SELECT PR, CAST(CONCAT(SUPP, '') AS VARCHAR) AS LIST, RN  FROM CTE1  WHERE RN = 1  UNION ALL  SELECT R.PR, CAST(CONCAT(R.LIST, ',' + C.SUPP) AS VARCHAR) AS LIST , C.RN  FROM RECURSIVECTE AS R  INNER JOIN CTE1 AS C ON C.RN = R.RN + 1 AND R.PR = C.PR  )  , REVERSEORDER AS  (  SELECT \*, ROW\_NUMBER() OVER(PARTITION BY PR ORDER BY RN DESC) AS REVERSEORDERING  FROM RECURSIVECTE  )  SELECT \*  FROM REVERSEORDER  WHERE REVERSEORDERING = 1  --CONCATENATE COLUMN VALUES TO A LIST USING XML PATH  --Note: SPECIAL CHARACTERS LIKE ‘<’ WILL BE ENCODED.  ;WITH CTE AS  (  SELECT \*  FROM (VALUES('LAPTOP', 'DELL'), ('LAPTOP', 'HP'), ('LAPTOP', 'COMPAQ'), ('MAC', 'APPLE')) AS P(PR, SUPP)  )  SELECT A.PR, STUFF((SELECT ',' + B.SUPP FROM CTE AS B WHERE B.PR = A.PR ORDER BY SUPP FOR XML PATH ('')), 1, 1, '')  FROM CTE AS A  GROUP BY PR;  --REQUIRES SQL SERVER 2017  ;WITH CTE AS  (  SELECT \*  FROM (VALUES('LAPTOP', 'DELL'), ('LAPTOP', 'HP'), ('LAPTOP', 'COMPAQ'), ('MAC', 'APPLE')) AS P(PR, SUPP)  )  SELECT A.PR, STRING\_AGG(SUPP, ',') WITHIN GROUP (ORDER BY SUPP ASC) AS List  FROM CTE AS A  GROUP BY PR; |