|  |  |
| --- | --- |
| **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. |  |

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