bite-sized.sql

SQL: INSERT SPACE BETWEEN FIRST AND LAST NAME

From

	names
1	HomerSimpson
2	MaggieSimpson
3	MargeSimpson
4	LisaSimpson
5	BartSimpson
6	BarneyGumble
7	RalphWiggum
8	SeymourSkinner
9	WaylonSmithers
10	MoeSzyslak

To

	full_name
1	Homer Simpson
2	Maggie Simpson
3	Marge Simpson
4	Lisa Simpson
5	Bart Simpson
6	Barney Gumble
7	Ralph Wiggum
8	Seymour Skinner
9	Waylon Smithers
10	Moe Szyslak

SQL Server: PATINDEX

SQL Server doesn't support regular expressions. The pattern used in PATINDEX is a pseudo-regex

PATINDEX returns the position the search string is found

"lower-case letter followed by an upper-case letter"

```
WITH position AS
(
SELECT
    names,
    PATINDEX(
        '%[a-z][A-Z]%',
        names COLLATE Latin1_General_BIN
        ) AS pos
FROM #no_space
)
SELECT
    names,
    STUFF(names, pos + 1, 0, ' ') AS full_name
FROM position;
```

Default collation is case-insensitive, so use a case-sensitive collation instead

At position pos + 1 in the text in names, remove 0 characters and insert (STUFF) a single space at that position

PostgreSQL: regexp_matches

regexp_matches returns an array of the substrings of the 1st parameter that match the regex in the 2nd. The function call is wrapped in parentheses to allow us to use array indexing to extract the result.

Regex break-down

^ = at the beginning

(.*)? = any characters

(?=[A-Z]) = followed by any upper-case letter. The upper-case letter is *not* included in the match. This is known as a positive look-ahead.

```
SELECT
    (regexp_matches(
        names,
        '^(.*)?(?=[A-Z])')
    )[1]
        || ' ' || (regexp_matches(
                        names,
                        '(?<=[a-z])[A-Z].*?$'
                    )[1] as full_name
FROM no_space;
```

Return the first array item (i.e. the substring matched by the expression)

Regex break-down

[A-Z] = An upper-case letter,

.*?$ = then all characters to the end of the string,

(?<=[a-z]) = following a lower-case letter. The lower-case letter is not part of the match. This is known as a positive look-behind.

MySQL: regexp_instr

MySQL doesn't support regex look-aheads or look-behinds

regexp_instr returns the position of the match

Param names highlighted

1) The expression to search
2) Regex pattern (a lower-case letter followed by an upper-case letter)
3) Starting position (1=start from beginning of string)
4) Match occurrence (1=first occurrence)
5) The return_option (1=return the position *after* the match substring)
6) The match_type (c=case sensitive)

```
WITH position
AS
(
SELECT
    names,
    regexp_instr(
      names, '[a-z][A-Z]', 1, 1, 1, 'c'
    ) - 1 as pos
FROM no_space
)
SELECT
    CONCAT(LEFT(names,pos-1),' ',MID(names,pos,10))
AS full_name
FROM position;
```

LEFT, MID and CONCAT work the same as in Excel