Challenge July-2025
Rules with Regular Expressions

A solution with DT5GL/SQL/Python by Jack Jansonius – 30 July 2025

Problem Statement (from the website):

A university is developing a system that helps a student to select certain courses to receive a desired degree. Each course has a unique code, such as "MA523" or "CS38251". Help a university create a simple decision service that evaluates a set of all courses to split it into two lists of allowed and not allowed courses, following these rules:

RULE 1. If the Degree Code is "MA Single" and the Course Code is similar to MA52# or MA62#, where # stands for any single character, add this course to the list of allowed courses.

RULE 2. If the Degree Code is not "CS%" and the Course Code is similar to CS%288% or CS%289%, where % stands for any combination of characters, add this course to the list of not allowed courses.

Testdata in the database:

SQLite/PostgreSQL/Etc.:

Table: course

	degree code	course code
1	MA Single	MA52
2	MA Single	MA521
3	MA Single	MA5212
4	MA Single	MA621
5	MA Singl	MA621
6	MA	CS288
7	MA	cs288
8	MA	CS1288
9	MA	CS2881
10	MA	CS1228834
11	MA	CS289
12	MA	CS992899
13	CS	CS288
14	CS2	CS288
15	CS22	CS288

Implementation of the decision model in DT5GL version 1:

```
SQLite_database: "Database/courses.sqlite3"
# PostgreSQL_database: "courses"

rTable 0:
If:                                                | 0| 1| 2| 3| 4| 5|
'Next Course'                                      | Y| Y| Y| Y| Y| N|
course.degree_code = "MA Single"                   | Y| Y| -| -| -| -|
true = like(course.course_code, "MA52#")           | Y| -| -| -| -| -|
true = like(course.course_code, "MA62#")           | -| Y| -| -| -| -|
true = like(course.degree_code, "CS%")             | -| -| N| N| -| -|
true = like(course.course_code, "CS%288%")         | -| -| Y| -| -| -|
true = like(course.course_code, "CS%289%")         | -| -| -| Y| -| -|
Then:
course is allowed                                  | X| X|  |  |  |  |
course is not_allowed                              |  |  | X| X|  |  |
course is not_classified                           |  |  |  |  | X|  |
course is finished                                 |  |  |  |  |  | X|
# .......

Proposition: 'Next Course'
Obtain_instance_from_database_view: course

Database_view: course
With_attributes: degree_code, course_code
Query:
 SELECT * FROM course
 LIMIT 1 OFFSET %s
With_arguments: course.auto_index

Initial_instructions:
>>: true = 1
>>: allowed_courses = ""
>>: not_allowed_courses = ""
>>: not_classified_courses = ""
End_Instructions

Attribute: course.degree_code    Type: Text
Attribute: course.course_code    Type: Text
Attribute: found_course          Type: Text
Equals: ", (" + course.degree_code + "," + course.course_code + ")"

# discard leading ", " if courses are found
Attribute: print_allowed         Type: Text
Equals: allowed_courses[2:] if allowed_courses else "None"
Attribute: print_not_allowed     Type: Text
Equals: not_allowed_courses[2:] if not_allowed_courses else "None"
Attribute: print_not_classified  Type: Text
Equals: not_classified_courses[2:] if not_classified_courses else "None"

GOALATTRIBUTE: course
Repeat_until: finished

Case: finished
Print: "Allowed courses: %s. " print_allowed
Print: "Not allowed courses: %s. " print_not_allowed
Print: "Not classified courses: %s. " print_not_classified

Case: allowed
>>: allowed_courses = allowed_courses + found_course

Case: not_allowed
>>: not_allowed_courses = not_allowed_courses + found_course

Case: not_classified
>>: not_classified_courses = not_classified_courses + found_course
```

Added to DTFunctions.py[1]:

```
def like(text, pattern):
    """
    Implements wildcard matching similar to SQL LIKE operator.
    Supports % for any sequence of characters and # for any single character.

    Args:
        text (str): The text to match against
        pattern (str): The wildcard pattern

    Returns:
        bool: True if text matches the pattern
    """
    # Escape regex special characters except our wildcards
    pattern = re.escape(pattern.replace('%', '\x00').replace('#', '\x01'))
    pattern = pattern.replace('\x00', '.*').replace('\x01', '.')

    # Match entire string
    return bool(re.fullmatch(pattern, text))
```

Testrun

PS C:\Users\jjans\dt5gl2504> .\dt.exe[2] -source:Wildcards_v1.txt -nti
Allowed courses: (MA Single,MA521), (MA Single,MA621).
Not allowed courses: (MA,CS288), (MA,CS1288), (MA,CS2881), (MA,CS1228834), (MA,CS289), (MA,CS992899).
Not classified courses: (MA Single,MA52), (MA Single,MA5212), (MA Singl,MA621), (MA,cs288), (CS,CS288), (CS2,CS288), (CS22,CS288).
Time elapsed: 0:00:00.069082

Version 2:
PS C:\Users\jjans\dt5gl2504> .\dt.exe -source:Wildcards_v2.txt -nti
Allowed courses: (MA Single,MA521), (MA Single,MA621).
Not allowed courses: (MA,CS288), (MA,CS1288), (MA,CS2881), (MA,CS1228834), (MA,CS289), (MA,CS992899).
Not classified courses: (MA Single,MA52), (MA Single,MA5212), (MA Singl,MA621), (MA,cs288), (CS,CS288), (CS2,CS288), (CS22,CS288).
Time elapsed: 0:00:00.069104

[1] Although I am a fan of ChatGPT, DeepSeek came up with a working solution a little faster.
2 DT.exe is Python code, compiled to C, and runs directly under Windows (without pre-installation of Python). Download and all necessary files available at: https://github.com/JackJansonius/DT5GL

Version 2:

Same as version 1, but[3]:

```
rTable 0:
If:                              | 0| 1| 2| 3| 4| 5|
'Next Course'                    | Y| Y| Y| Y| Y| N|
Degree_Code = "MA Single"        | Y| Y| -| -| -| -|
Course_Code = "MA52#"            | Y| -| -| -| -| -|
Course_Code = "MA62#"            | -| Y| -| -| -| -|
Degree_Code = "CS%"              | -| -| N| N| -| -|
Course_Code = "CS%288%"          | -| -| Y| -| -| -|
Course_Code = "CS%289%"          | -| -| -| Y| -| -|
Then:
course is allowed                | X| X|  |  |  |  |
course is not_allowed            |  |  | X| X|  |  |
course is not_classified         |  |  |  |  | X|  |
course is finished               |  |  |  |  |  | X|
# .......
#

Attribute: Degree_Code
Equals: "MA Single" if course.degree_code == "MA Single"  else  \
        "CS%"       if like(course.degree_code, "CS%")    else  \
        ""

Attribute: Course_Code
Equals: "MA52#"     if like(course.course_code, "MA52#")    else  \
        "MA62#"     if like(course.course_code, "MA62#")    else  \
        "CS%288%"   if like(course.course_code, "CS%288%")  else  \
        "CS%289%"   if like(course.course_code, "CS%289%")  else  \
        ""
```

[3] It needs no further explanation that this version was inspired by Jacob Feldman's solution.

SQL code to create the databases (SQLite/PostgreSQL)[4]

```
-- DDL generated by DBeaver
CREATE TABLE course (
degree_code TEXT,
course_code TEXT
);


INSERT INTO course (degree_code,course_code) VALUES
        ('MA Single','MA52'),
        ('MA Single','MA521'),
        ('MA Single','MA5212'),
        ('MA Single','MA621'),
        ('MA Singl','MA621'),
        ('MA','CS288'),
        ('MA','cs288'),
        ('MA','CS1288'),
        ('MA','CS2881'),
        ('MA','CS1228834'),
        ('MA','CS289'),
        ('MA','CS992899'),
        ('CS','CS288'),
        ('CS2','CS288'),
        ('CS22','CS288');
```

[4] Tables were created with SQLiteStudio (https://sqlitestudio.pl/); the related SQL instructions generated with DBeaver.