Challenge April-2025
Case Assignments

A solution with DT5GL/SQL by Jack Jansonius – 26 April 2025

Problem Statement (from the website):

An analytical firm assigns different cases to its analysts using the following rules:

- Case must be assigned to an analyst who has the same focus area as the case type
- Analysts can not work on a new case with an amount higher than their maximum allowed case amount.
- Analysts can not work on a new case if it puts them over their maximum total cases dollar amount
- Analyst Levels must correspond to Case Complexity: analysts can work only on new cases with complexity between their Minimum Case Complexity and Maximum Case Complexity (inclusive) as described in the following table:

Qualification Level	Minimum Case Complexity	Maximum Case Complexity
1	1	2
2	1	2
3	1	2
4	1	3
5	2	3
6	2	4
7	2	4
8	3	4
9	3	5
10	4	5

Here is the list of analysts with their current load:

Analyst Name	Analyst Level	Total Amount of Assigned Cases	Total Number of Assigned Cases	Analyst Focus Areas	Maximum Allowable Case Amount	Total Cases Maximum Dollar Amount
Tim Smith	10	$35,000,000	12	Technology Research Construction	$50,000,000	$75,000,000
Sue Rogers	5	$5,000,000	10	Technology Research	$1,000,000	$7,000,000
Sam Howard	8	$19,000,000	9	Construction Research	$20,000,000	$20,000,000
Jill Ryan	9	$14,700,000	10	Technology Research Construction	$1,500,000	$25,000,000
Debbie Smith	6	$8,000,000	14	Research Technology	$10,000,000	$10,000,000
Debbie Bowers	7	$6,000,000	8	Technology Research	$1,000,000	$7,500,000
Kevin Jones	4	$2,800,000	8	Research Technology	$3,000,000	$3,000,000
Roger Howland	2	$850,000	7	Construction Technology	$300,000	$1,000,000

Here is the list of new cases:

Case Number	Case Amount	Case Complexity	Case Type
112	$50,000	3	Technology
113	200,000	1	Technology
114	1,500,000	4	Construction
115	300,000	4	Research

The firm needs to decide which analysts should be assigned to these cases. If there are multiple options, the firm prefers to minimize the overqualification when analysts are assigned to the cases below their levels. Can you create a decision service capable of addressing this and similar problems?

Tables in the database:

SQLite/PostgreSQL/Etc.:

Table: analyst

id	name	level	amount_assigned	number_assigned	max_amount_per_case	max_total_amount
1	Tim Smith	10	35000000	12	50000000	75000000
2	Sue Rogers	5	5000000	10	1000000	7000000
3	Sam Howard	8	19000000	9	20000000	20000000
4	Jill Ryan	9	14700000	10	1500000	25000000
5	Debbie Smith	6	8000000	14	10000000	10000000
6	Debbie Bowers	7	6000000	8	1000000	7500000
7	Kevin Jones	4	2800000	8	3000000	3000000
8	Roger Howland	2	850000	7	300000	1000000

Table: analyst-area

analyst_id	area_id
1	1
1	2
1	3
2	1
2	2
3	2
3	3
4	1
4	2
4	3
5	1
5	2
6	1
6	2
7	1
7	2
8	1
8	3

Table: area

id	name
1	Technology
2	Research
3	Construction

Table: cases

id	amount	complexity	area_id
112	50000	3	1
113	200000	1	1
114	1500000	4	3
115	300000	4	2

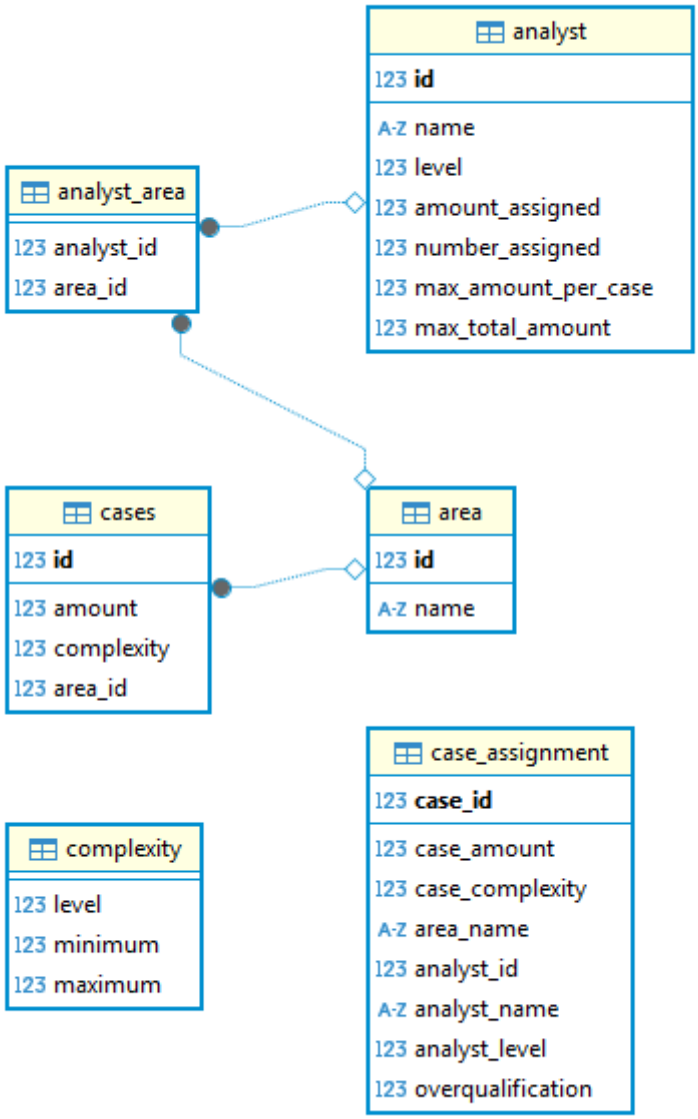
Table: complexity

level	minimum	maximum
1	1	2
2	1	2
3	1	2
4	1	3
5	2	3
6	2	4
7	2	4
8	3	4
9	3	5
10	4	5

Table: case_assignment (after testrun)

case_id	case_amount	case_complexity	area_name	analyst_id	analyst_name	analyst_level	overqualification
112	50000	3	Technology	2	Sue Rogers	5	2
113	200000	1	Technology	7	Kevin Jones	4	3
114	1500000	4	Construction	4	Jill Ryan	9	5
115	300000	4	Research	5	Debbie Smith	6	2

ER-schema (DBeaver)



Implementation of the decision model in DT5GL/SQL:

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


Table 0:
If:                                          | 0| 1|
'Next Case'                                  | Y| N|
Then:
case is selected                             | X|  |
case is finished                             |  | X|
# .......


rTable 1:
If:                                                                  | 0| 1| 2| 3|
'Next Analyst'                                                       | Y| Y| N| N|
'Case Area in Analyst Focus Areas'                                   | Y| -| -| -|
case.amount <= analyst.max_amount_per_case                           | Y| -| -| -|
case.amount <= analyst.max_total_amount - analyst.amount_assigned    | Y| -| -| -|
case.complexity >= analystlevel.minimum                              | Y| -| -| -|
case.complexity <= analystlevel.maximum                              | Y| -| -| -|
current_overqualification > analyst.level - case.complexity          | Y| -| -| -|
current_analyst_name = ""                                            | -| -| Y| N|
Then:
analyst is swap                                                      | X|  |  |  |
analyst is noswap                                                    |  | X|  |  |
analyst is notfound                                                  |  |  | X|  |
analyst is found                                                     |  |  |  | X|
# .......

Proposition: 'Next Case'
Obtain_instance_from_database_view: case

Proposition: 'Next Analyst'
Obtain_instance_from_database_view: analyst

Proposition: 'Case Area in Analyst Focus Areas'
Obtain_instance_from_database_view: samefocusarea


Database_view: case
With_attributes: id, amount, complexity, area_id
Query:
 SELECT * FROM cases ORDER BY complexity ASC, amount ASC
 LIMIT 1 OFFSET %s
With_arguments: case.auto_index


Database_view: analyst
With_attributes: id, name, level, amount_assigned, number_assigned,
max_amount_per_case, max_total_amount
Query:
 SELECT * FROM analyst ORDER BY id ASC
 LIMIT 1 OFFSET %s
With_arguments: analyst.auto_index
```

```
Database_view: samefocusarea
With_attributes: id
Query:
SELECT area.id
FROM cases
     JOIN area ON cases.area_id = area.id
     JOIN analyst_area ON area.id = analyst_area.area_id
WHERE cases.id= %s AND analyst_area.analyst_id= %s
With_arguments: case.id, analyst.id


Database_view: analystlevel
With_attributes: minimum, maximum
Query:
 SELECT minimum, maximum FROM complexity
 WHERE level = %s
With_arguments: analyst.level


Database_view: case_area
With_attributes: name
Query:
 SELECT name FROM area
 WHERE id = %s
With_arguments: case.area_id


Attribute: analyst.max_amount_per_case  Type: Integer
Attribute: analyst.max_total_amount     Type: Integer
Attribute: analyst.amount_assigned      Type: Integer
Attribute: analystlevel.maximum         Type: Integer
Attribute: analystlevel.minimum         Type: Integer
Attribute: analyst.level                Type: Integer
Attribute: analyst.id                   Type: Integer
Attribute: analyst.name                 Type: Text
Attribute: case.amount                  Type: Integer
Attribute: case.complexity              Type: Integer


Initial_instructions:
>>: total_overqualification = 0
End_Instructions

Initial_database_setup: delete case assignements
Query:
   DELETE FROM case_assignment
End_Query
```

```
GOALATTRIBUTE: case
Repeat_until: finished

Case: finished
Print: "Total overqualification: %s. " total_overqualification

Case: selected
>>: current_analyst_id = 0
>>: current_analyst_name = ""
>>: current_analyst_level = 0
>>: current_overqualification = 999


GOALATTRIBUTE: analyst
Repeat_until: notfound, found

Case: swap
>>: current_analyst_id = analyst.id
>>: current_analyst_name = analyst.name
>>: current_analyst_level = analyst.level
>>: current_overqualification = analyst.level - case.complexity

Case: noswap
Print: "#REM# - Do nothing"

Case: notfound
Print: "No analyst found for case: %s." case.id

Case: found
Print: "Analyst %s. %s assigned to case %s with overqualification: %s"
current_analyst_id current_analyst_name case.id current_overqualification
>SQL:  "UPDATE analyst "
-SQL:  "   SET amount_assigned = amount_assigned + %s, "  case.amount
-SQL:  "       number_assigned = number_assigned + 1 "
<SQL:  " WHERE id = %s "                                  current_analyst_id

>SQL:  "INSERT INTO case_assignment "
-SQL:  "(case_id, case_amount, case_complexity, area_name, analyst_id,
analyst_name, analyst_level, overqualification) "
-SQL:  "VALUES (%s, "  case.id
-SQL:  "%s, " case.amount
-SQL:  "%s, " case.complexity
-SQL:  " '%s', " case_area.name
-SQL:  "%s, " current_analyst_id
-SQL:  " '%s', " current_analyst_name
-SQL:  "%s, " current_analyst_level
<SQL:  "%s) " current_overqualification

>>: total_overqualification = total_overqualification + current_overqualification
```

Testrun:

PS C:\Users\jjans\dt5gl2504> .\dt.exe[1] -source:AnalystCase.txt -nti
Analyst 7. Kevin Jones assigned to case 113 with overqualification: 3
Analyst 2. Sue Rogers assigned to case 112 with overqualification: 2
Analyst 5. Debbie Smith assigned to case 115 with overqualification: 2
Analyst 4. Jill Ryan assigned to case 114 with overqualification: 5
Total overqualification: 12.
Time elapsed: 0:00:00.057704

Table: case_assignment (after testrun)

case_id	case_amount	case_complexity	area_name	analyst_id	analyst_name	analyst_level	overqualification
112	50000	3	Technology	2	Sue Rogers	5	2
113	200000	1	Technology	7	Kevin Jones	4	3
114	1500000	4	Construction	4	Jill Ryan	9	5
115	300000	4	Research	5	Debbie Smith	6	2

Table: analyst (after testrun)

id	name	level	amount_assigned	number_assigned	max_amount	max_total_amount
1	Tim Smith	10	35000000	12	50000000	75000000
2	Sue Rogers	5	5050000	11	1000000	7000000
3	Sam Howard	8	19000000	9	20000000	20000000
4	Jill Ryan	9	16200000	11	1500000	25000000
5	Debbie Smith	6	8300000	15	10000000	10000000
6	Debbie Bowers	7	6000000	8	1000000	7500000
7	Kevin Jones	4	3000000	9	3000000	3000000
8	Roger Howland	2	850000	7	300000	1000000

[1] 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

SQL code to create the databases (SQLite/PostgreSQL)[2]

```
CREATE TABLE analyst (
    id                  INTEGER PRIMARY KEY,
    name                TEXT,
    level               INTEGER,
    amount_assigned     INTEGER,
    number_assigned     INTEGER,
    max_amount_per_case INTEGER,
    max_total_amount    INTEGER
);

INSERT INTO analyst
(id,name,"level",amount_assigned,number_assigned,max_amount_per_case,max_total_amou
nt)
VALUES
        (1,'Tim Smith',10,35000000,12,50000000,75000000),
        (2,'Sue Rogers',5,5000000,10,1000000,7000000),
        (3,'Sam Howard',8,19000000,9,20000000,20000000),
        (4,'Jill Ryan',9,14700000,10,1500000,25000000),
        (5,'Debbie Smith',6,8000000,14,10000000,10000000),
        (6,'Debbie Bowers',7,6000000,8,1000000,7500000),
        (7,'Kevin Jones',4,2800000,8,3000000,3000000),
        (8,'Roger Howland',2,850000,7,300000,1000000);


CREATE TABLE area (
    id   INTEGER PRIMARY KEY,
    name TEXT
);

INSERT INTO area (id, name)
VALUES
     (1, 'Technology'),
     (2, 'Research'),
     (3, 'Construction');


CREATE TABLE analyst_area (
    analyst_id INTEGER REFERENCES analyst (id),
    area_id    INTEGER REFERENCES area (id)
);

INSERT INTO analyst_area (analyst_id,area_id)
VALUES
        (1,1), (1,2), (1,3),
        (2,1), (2,2),
        (3,2), (3,3),
        (4,1), (4,2), (4,3),
        (5,1), (5,2),
        (6,1), (6,2),
        (7,1), (7,2),
        (8,1), (8,3);
```

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

```
CREATE TABLE cases (
    id         INTEGER PRIMARY KEY,
    amount     INTEGER,
    complexity INTEGER,
    area_id    INTEGER REFERENCES area (id)
);


INSERT INTO cases (id,amount,complexity,area_id)
VALUES
        (112,50000,3,1),
        (113,200000,1,1),
        (114,1500000,4,3),
        (115,300000,4,2);


CREATE TABLE complexity (
    level   INTEGER,
    minimum INTEGER,
    maximum INTEGER
);

INSERT INTO complexity ("level",minimum,maximum)
VALUES
        (1,1,2),
        (2,1,2),
        (3,1,2),
        (4,1,3),
        (5,2,3),
        (6,2,4),
        (7,2,4),
        (8,3,4),
        (9,3,5),
        (10,4,5);


CREATE TABLE case_assignment[3] (
    case_id         INTEGER PRIMARY KEY,
    case_amount     INTEGER,
    case_complexity INTEGER,
    area_name       TEXT,
    analyst_id      INTEGER,
    analyst_name    TEXT,
    analyst_level   INTEGER,
    overqualification INTEGER
);
```

[3] This table was generated by ChatGPT based on all previous tables and is also used in the final solution in Appendix A..

Appendix A: a solution with PostgreSQL (ChatGPT)

```
DO $$
DECLARE
    case_rec RECORD;
    chosen_analyst RECORD;
BEGIN
    -- Step 1: Clear any previous assignments
    DELETE FROM case_assignment;

    -- Step 2: Process cases starting with the lowest complexity
    FOR case_rec IN
        SELECT * FROM newcase ORDER BY complexity ASC, amount ASC
    LOOP
        -- Step 3: Find the best eligible analyst for the case
        SELECT
            a.id AS analyst_id,
            e.name AS analyst_name,
            a.level AS analyst_level,
            a.amount_assigned,
            a.max_total_amount,
            a.max_amount_per_case,
            GREATEST(a.level - case_rec.complexity, 0) AS overqualification
        INTO chosen_analyst
        FROM analist a
        JOIN employee e ON e.id = a.id
        JOIN complexity c ON c.level = a.level
        JOIN analist_areas aa ON aa.analist_id = a.id
        WHERE
            aa.area_id = case_rec.area_id
            AND case_rec.amount <= a.max_amount_per_case
            AND (a.amount_assigned + case_rec.amount) <= a.max_total_amount
            AND case_rec.complexity BETWEEN c.minimum AND c.maximum
        ORDER BY overqualification ASC, a.level ASC
        LIMIT 1;

     -- Step 4: If a match is found, assign the case and update the analyst's total
        IF FOUND THEN
            INSERT INTO case_assignment (
                case_id, case_amount, case_complexity, area_name,
                analyst_id, analyst_name, analyst_level, overqualification
            )
            VALUES (
                case_rec.id,
                case_rec.amount,
                case_rec.complexity,
                (SELECT name FROM area WHERE id = case_rec.area_id),
                chosen_analyst.analyst_id,
                chosen_analyst.analyst_name,
                chosen_analyst.analyst_level,
                GREATEST(chosen_analyst.analyst_level - case_rec.complexity, 0)
            );

            -- Update the analyst's assigned amount
            UPDATE analist
            SET amount_assigned = amount_assigned + case_rec.amount
            WHERE id = chosen_analyst.analyst_id;
        END IF;
    END LOOP;
END $$;
```

```
112    50000        3      "Technology"        2      "Sue Rogers"        5      2
113    200000       1      "Technology"        7      "Kevin Jones"       4      3
114    1500000      4      "Construction"      4      "Jill Ryan"         9      5
115    300000       4      "Research"          5      "Debbie Smith"      6      2
```