Julia Ruszer 247775

Dominik Gałkowski 247659

Jan Śladowski 247806

Wiktor Żelechowski 247833

**Zadanie 1**

Utwórz funkcję, która zwróci liczbę pracowników zatrudnionych na określonym stanowisku w określonym dziale w określonym kraju. Nazwa stanowiska, nazwa działu i nazwa kraju są przekazywane do funkcji jako parametry. Wywołaj funkcję z parametrami Sales Manager, Sales i United Kingdom.

**PostgreSQL**

CREATE OR REPLACE FUNCTION count\_employees(

job\_title\_fun VARCHAR(35),

department\_name\_fun VARCHAR(30),

country\_name\_fun VARCHAR(40)

) RETURNS INTEGER AS $$

BEGIN

RETURN (

SELECT COUNT(e.employee\_id)

FROM employees e

JOIN jobs j ON e.job\_id = j.job\_id

JOIN departments d ON e.department\_id = d.department\_id

JOIN locations l ON d.location\_id = l.location\_id

JOIN countries c ON l.country\_id = c.country\_id

WHERE j.job\_title = job\_title\_fun

AND d.department\_name = department\_name\_fun

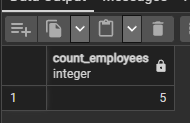
AND c.country\_name = country\_name\_fun

);

END;

$$ LANGUAGE plpgsql;

SELECT count\_employees('Sales Manager', 'Sales', 'United Kingdom');

****

**MS SQL Server**

CREATE OR ALTER FUNCTION func (@input1 NVARCHAR(100), @input2 NVARCHAR(100), @input3 NVARCHAR(100))

RETURNS INT

AS BEGIN

DECLARE @employeeCount INT

SELECT @employeeCount = COUNT(\*) from employees e

JOIN departments d ON d.department\_id = e.department\_id

JOIN locations l ON l.location\_id = d.location\_id

JOIN countries c ON c.country\_id = l.country\_id

JOIN jobs j ON j.job\_id = e.job\_id

WHERE j.job\_title = @input1

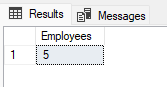
AND d.department\_name = @input2

AND c.country\_name = @input3;

RETURN @employeeCount;

END

SELECT dbo.func('Sales Manager', 'Sales', 'United Kingdom') AS Employees

****

**Zadanie 2**  
  
Utwórz wyzwalacz, który przy zmianie stanowiska danego pracownika:  
– zaktualizuje jego datę zatrudnienia w tabeli employees na dzień jutrzejszy,  
– zarchiwizuje dane o jego poprzednim stanowisku, tzn. doda odpowiednie informacje do tabeli job\_history i ustawi datę końcową na dzień dzisiejszy,  
– sprawdzi, czy jego aktualne wynagrodzenie należy do zdefiniowanego przedziału wartości wynagrodzeń dla jego nowego stanowiska. Jeżeli jego aktualne wynagrodzenie jest niższe niż minimalna wartość przedziału, to zostanie zaktualizowane do tejże wartości. Dodatkowo zostanie wypisana informacja o imieniu i nazwisku pracownika oraz kwocie jego podwyżki (różnicy pomiędzy nową i starą kwotą wynagrodzenia). Jeżeli jego aktualne wynagrodzenie jest wyższe niż maksymalna wartość przedziału, to maksymalne wynagrodzenie dla jego nowego stanowiska zostanie zaktualizowane do wartości jego aktualnego wynagrodzenia. Potwierdź działanie dla wszystkich przypadków testowych. *Uwaga!* Wyzwalacz powinien pracować także przy zmianie stanowiska dla wielu pracowników jednocześnie.

**PostgreSQL**

CREATE OR REPLACE FUNCTION trigger\_employee\_job\_change()

RETURNS TRIGGER AS $$

DECLARE

new\_min\_salary NUMERIC(6);

new\_max\_salary NUMERIC(6);

current\_salary NUMERIC(6);

salary\_increase NUMERIC;

BEGIN

SELECT min\_salary, max\_salary INTO new\_min\_salary, new\_max\_salary

FROM jobs

WHERE job\_id = NEW.job\_id;

NEW.hire\_date := CURRENT\_DATE + INTERVAL '1 day';

INSERT INTO job\_history (employee\_id, start\_date, end\_date, job\_id, department\_id)

VALUES (OLD.employee\_id, OLD.hire\_date, CURRENT\_DATE, OLD.job\_id, OLD.department\_id);

current\_salary := NEW.salary;

--jak mniejsze to podnoisi pracownikowi

IF current\_salary < new\_min\_salary THEN

salary\_increase := new\_min\_salary - current\_salary;

NEW.salary := new\_min\_salary;

RAISE NOTICE 'Pracownik % % otrzymał podwyżkę o kwotę %', NEW.first\_name, NEW.last\_name, salary\_increase;

--jak wieksze to zmienia dla stanowiska

ELSIF current\_salary > new\_max\_salary THEN

UPDATE jobs

SET max\_salary = current\_salary

WHERE job\_id = NEW.job\_id;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER job\_change\_trigger

BEFORE UPDATE OF job\_id ON employees

FOR EACH ROW

WHEN (OLD.job\_id IS DISTINCT FROM NEW.job\_id)

EXECUTE FUNCTION trigger\_employee\_job\_change();

--dobre wynagrodzenie

INSERT INTO employees (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

VALUES (1000, 'imie1', 'nazwisko1', 'test1', '515.123.1234', '2000-01-01', 'IT\_PROG', 11000, NULL, NULL, 60);

--wyzsze wynagrodzenie

INSERT INTO employees (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

VALUES (1001, 'imie2', 'nazwisko2', 'test2', '515.123.1234', '2000-01-01', 'IT\_PROG', 25000, NULL, NULL, 60);

--za niskie wynagrodzenie

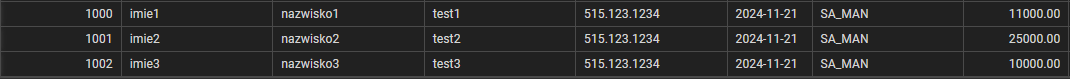
INSERT INTO employees (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

VALUES (1002, 'imie3', 'nazwisko3', 'test3', '515.123.1234', '2000-01-01', 'IT\_PROG', 5000, NULL, NULL, 60);

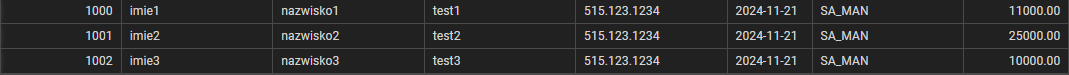
UPDATE employees SET job\_id = 'SA\_MAN' WHERE employee\_id = 1000;

UPDATE employees SET job\_id = 'SA\_MAN' WHERE employee\_id = 1001;

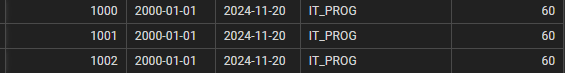
UPDATE employees SET job\_id = 'SA\_MAN' WHERE employee\_id = 1002;

****

****

****

****

****

**MS SQL Server**

CREATE OR ALTER TRIGGER trig

ON employees

AFTER UPDATE

AS

BEGIN

INSERT INTO job\_history(employee\_id, start\_date, end\_date, job\_id, department\_id)

SELECT

d.employee\_id,

d.hire\_date,

CAST(GETDATE() AS DATE),

d.job\_id,

d.department\_id

FROM deleted d

JOIN inserted i on d.employee\_id = i.employee\_id

WHERE

i.job\_id <> d.job\_id;

UPDATE employees

SET hire\_date = CAST(DATEADD(DAY, 1, GETDATE()) AS DATE)

FROM inserted i

WHERE employees.employee\_id = i.employee\_id

AND i.job\_id <> (SELECT job\_id FROM deleted WHERE deleted.employee\_id = i.employee\_id);

DECLARE @new\_min\_salary NUMERIC(8,2), @new\_max\_salary NUMERIC(8,2), @current\_salary NUMERIC(8,2), @salary\_increase NUMERIC(8,2);

DECLARE @employee\_id NUMERIC(6);

DECLARE @first\_name VARCHAR(20), @last\_name VARCHAR(20), @job\_id VARCHAR(20);

DECLARE salary\_cursor CURSOR FOR

SELECT i.salary, j.min\_salary, j.max\_salary, i.employee\_id, i.first\_name, i.last\_name, i.job\_id

FROM inserted i

JOIN jobs j ON i.job\_id = j.job\_id

WHERE i.job\_id <> (SELECT job\_id FROM deleted WHERE deleted.employee\_id = i.employee\_id);

OPEN salary\_cursor

FETCH NEXT FROM salary\_cursor INTO @current\_salary, @new\_min\_salary, @new\_max\_salary, @employee\_id, @first\_name, @last\_name, @job\_id;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF @current\_salary < @new\_min\_salary

BEGIN

SET @salary\_increase = @new\_min\_salary - @current\_salary

UPDATE employees

SET salary = @new\_min\_salary

WHERE employee\_id = @employee\_id

PRINT 'Pracownik ' + @first\_name + ' ' + @last\_name + ' otrzymał oodwyżkę o ' + CAST(@current\_salary AS VARCHAR(10));

END

ELSE IF @current\_salary > @new\_max\_salary

BEGIN

UPDATE jobs

SET max\_salary = @current\_salary

WHERE job\_id = @job\_id

END

FETCH NEXT FROM salary\_cursor INTO @current\_salary, @new\_min\_salary, @new\_max\_salary, @employee\_id, @first\_name, @last\_name, @job\_id;

END;

CLOSE salary\_cursor

DEALLOCATE salary\_cursor

END

--dobre wynagrodzenie

INSERT INTO employees (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

VALUES (1000, 'imie1', 'nazwisko1', 'test1', '515.123.1234', '2000-01-01', 'IT\_PROG', 11000, NULL, NULL, 60);

--wyzsze wynagrodzenie

INSERT INTO employees (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

VALUES (1001, 'imie2', 'nazwisko2', 'test2', '515.123.1234', '2000-01-01', 'IT\_PROG', 28000, NULL, NULL, 60);

--za niskie wynagrodzenie

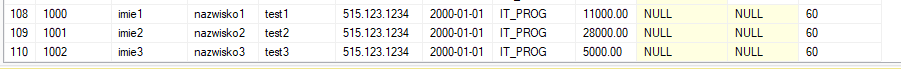
INSERT INTO employees (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

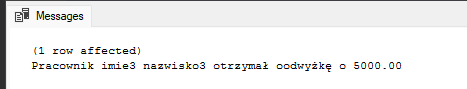
VALUES (1002, 'imie3', 'nazwisko3', 'test3', '515.123.1234', '2000-01-01', 'IT\_PROG', 5000, NULL, NULL, 60);

UPDATE employees SET job\_id = 'SA\_MAN' WHERE employee\_id = 1000;

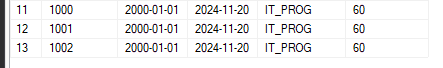
UPDATE employees SET job\_id = 'SA\_MAN' WHERE employee\_id = 1001;

UPDATE employees SET job\_id = 'SA\_MAN' WHERE employee\_id = 1002;





****

****

****

**Zadanie 3**  
  
Utwórz procedurę, która zmieni stanowisko na podane u pracowników zatrudnionych na określonym stanowisku w określonym kraju i poprzez parametr wyjściowy zwróci liczbę zmodyfikowanych rekordów oraz wyświetli id, imię, nazwisko i nazwę departamentu pracowników, których stanowisko zostało zmienione. Dodatkowo wypisze informacje o wszystkich departamentach z danego kraju razem z listą ich pracowników (id, imię i nazwisko), których stanowisko zostało zmienione. Jeżeli w jakimś departamencie w danym kraju nie pracuje żaden pracownik na danym stanowisku, wywoła jak najwyższy priorytetowo komunikat, który nie przerwie wykonywania kodu i wypisze: "Brak pracowników na stanowisku X w departamencie Y w kraju Z!", gdzie X jest nazwą poprzedniego stanowiska, Y jest nazwą aktualnie sprawdzanego departamentu, Z jest nazwą podanego kraju. Procedura ma także weryfikować podane dane. Jeżeli podany kraj nie istnieje, wywoła wyjątek, który przerwie wykonywanie kodu i wypisze: "Brak kraju X!", gdzie X to nazwa podanego kraju. Jeżeli w podanym kraju nie ma żadnego departamentu, wywoła wyjątek, który przerwie wykonywanie kodu i wypisze: "Brak departamentów w kraju X!", gdzie X to nazwa podanego kraju. Jeżeli w podanym kraju nie pracuje żaden pracownik, wywoła wyjątek, który przerwie wykonywanie kodu i wypisze: "Brak pracowników zatrudnionych w kraju X!", gdzie X to nazwa podanego kraju. W swoim rozwiązaniu wykorzystaj funkcję z zadania 1 oraz wyzwalacz z zadania 2. Wywołaj procedurę z nazwami stanowisk Sales Manager i Sales Representative oraz odpowiednią nazwą kraju, żeby przetestować wszystkie możliwe przypadki.

**PostgreSQL**

CREATE OR REPLACE PROCEDURE change\_job\_title\_procedure(

IN current\_job\_title VARCHAR(100),

IN new\_job\_title VARCHAR(100),

IN country\_name\_param VARCHAR(100),

OUT employee\_count INTEGER

)

LANGUAGE plpgsql

AS $$

DECLARE

department\_name\_var VARCHAR(100);

employee\_id NUMERIC(6, 0);

first\_name VARCHAR(100);

last\_name VARCHAR(100);

department\_id\_var VARCHAR(100);

dupa NUMERIC(6, 0);

BEGIN

IF NOT EXISTS (

SELECT 1 FROM countries c WHERE c.country\_name = country\_name\_param

) THEN

RAISE EXCEPTION 'Brak kraju %!', country\_name\_param;

END IF;

IF NOT EXISTS (

SELECT 1

FROM departments d

JOIN locations l ON d.location\_id = l.location\_id

JOIN countries c ON l.country\_id = c.country\_id

WHERE c.country\_name = country\_name\_param

) THEN

RAISE EXCEPTION 'Brak departamentów w kraju %!', country\_name\_param;

END IF;

IF NOT EXISTS (

SELECT 1

FROM employees e

JOIN departments d ON e.department\_id = d.department\_id

JOIN locations l ON d.location\_id = l.location\_id

JOIN countries c ON l.country\_id = c.country\_id

WHERE c.country\_name = country\_name\_param

) THEN

RAISE EXCEPTION 'Brak pracowników zatrudnionych w kraju %!', country\_name\_param;

END IF;

IF EXISTS (SELECT FROM pg\_tables WHERE tablename = 'temp\_updated\_employees') THEN

DROP TABLE temp\_updated\_employees;

END IF;

CREATE TEMP TABLE temp\_updated\_employees AS

SELECT

e.employee\_id,

e.first\_name,

e.last\_name,

d.department\_name,

new\_job\_title AS simulated\_job\_title

FROM employees e

JOIN jobs j ON e.job\_id = j.job\_id

JOIN departments d ON e.department\_id = d.department\_id

JOIN locations l ON d.location\_id = l.location\_id

JOIN countries c ON l.country\_id = c.country\_id

WHERE j.job\_title = current\_job\_title AND c.country\_name = country\_name\_param;

SELECT SUM(count\_employees(current\_job\_title, d.department\_name, country\_name\_param))

INTO employee\_count

FROM departments d

JOIN locations l ON d.location\_id = l.location\_id

JOIN countries c ON l.country\_id = c.country\_id

WHERE c.country\_name = country\_name\_param;

EXECUTE 'SELECT e.first\_name FROM employees e';

UPDATE employees e

SET job\_id = (

SELECT j\_new.job\_id

FROM jobs j\_new

WHERE j\_new.job\_title = new\_job\_title

)

WHERE e.employee\_id IN (

SELECT te.employee\_id

FROM temp\_updated\_employees te

);

FOR department\_name\_var IN

SELECT d.department\_name

FROM departments d

JOIN locations l ON d.location\_id = l.location\_id

JOIN countries c ON l.country\_id = c.country\_id

WHERE c.country\_name = country\_name\_param

LOOP

IF EXISTS (

SELECT 1

FROM temp\_updated\_employees te

WHERE te.department\_name = department\_name\_var

) THEN

RAISE NOTICE 'Departament: %', department\_name\_var;

FOR employee\_id, first\_name, last\_name IN

SELECT te.employee\_id, te.first\_name, te.last\_name

FROM temp\_updated\_employees te

WHERE te.department\_name = department\_name\_var

LOOP

RAISE NOTICE ' Employee ID: %, Name: % %', employee\_id, first\_name, last\_name;

END LOOP;

ELSE

RAISE NOTICE 'Brak pracowników na stanowisku % w departamencie % w kraju %!',

current\_job\_title, department\_name\_var, country\_name\_param;

END IF;

END LOOP;

END;

$$;

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

**Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie**

**Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie**

**Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie**

**MS SQL Server**

CREATE OR ALTER PROCEDURE changeJobTitle

@CurrentJobTitle NVARCHAR(100),

@NewJobTitle NVARCHAR(100),

@Country NVARCHAR(100),

@UpdatedCount INT OUTPUT

AS

BEGIN

SET NOCOUNT ON;

DECLARE @message NVARCHAR(max);

IF NOT EXISTS (

SELECT 1 FROM countries WHERE country\_name = @Country

)

BEGIN

SET @message = CONCAT('Brak kraju ', @Country, '!');

THROW 50001, @message, 1;

END;

IF NOT EXISTS (

SELECT 1

FROM departments d

JOIN locations l ON d.location\_id = l.location\_id

JOIN countries c ON l.country\_id = c.country\_id

WHERE c.country\_name = @Country

)

BEGIN

SET @message = CONCAT('Brak departamentów w kraju ', @Country, '!');

THROW 50002, @message, 1;

END;

IF NOT EXISTS (

SELECT 1

FROM employees e

JOIN departments d ON e.department\_id = d.department\_id

JOIN locations l ON d.location\_id = l.location\_id

JOIN countries c ON l.country\_id = c.country\_id

WHERE c.country\_name = @Country

)

BEGIN

SET @message = CONCAT('Brak pracowników zatrudnionych w kraju ', @Country, '!');

THROW 50003, @message, 1;

END;

SELECT @UpdatedCount = dbo.func(@CurrentJobTitle, d.department\_name, @Country)

FROM departments d

JOIN locations l ON d.location\_id = l.location\_id

JOIN countries c ON l.country\_id = c.country\_id

WHERE c.country\_name = @Country

CREATE TABLE #UpdatedEmployees (

employee\_id NUMERIC(6, 0),

first\_name NVARCHAR(100),

last\_name NVARCHAR(100),

department\_name NVARCHAR(100)

);

INSERT INTO #UpdatedEmployees (employee\_id, first\_name, last\_name, department\_name)

SELECT e.employee\_id, e.first\_name, e.last\_name, d.department\_name

FROM employees e

JOIN jobs j ON e.job\_id = j.job\_id

JOIN departments d ON e.department\_id = d.department\_id

JOIN locations l ON d.location\_id = l.location\_id

JOIN countries c ON l.country\_id = c.country\_id

WHERE j.job\_title = @CurrentJobTitle AND c.country\_name = @Country;

UPDATE employees

SET job\_id = (SELECT job\_id FROM jobs WHERE job\_title = @NewJobTitle)

WHERE employee\_id IN (SELECT employee\_id FROM #UpdatedEmployees);

SELECT employee\_id, first\_name, last\_name, department\_name

FROM #UpdatedEmployees;

DECLARE @msg NVARCHAR(MAX) = '';

PRINT 'Informacje o departamentach i pracownikach:';

DECLARE @DepartmentName NVARCHAR(100);

DECLARE departmentCursor CURSOR FOR

SELECT d.department\_name

FROM departments d

JOIN locations l ON d.location\_id = l.location\_id

JOIN countries c ON l.country\_id = c.country\_id

WHERE c.country\_name = @Country;

OPEN departmentCursor;

FETCH NEXT FROM departmentCursor INTO @DepartmentName;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF EXISTS (

SELECT 1

FROM #UpdatedEmployees ue

JOIN employees e ON ue.employee\_id = e.employee\_id

JOIN departments d ON e.department\_id = d.department\_id

WHERE d.department\_name = @DepartmentName

)

BEGIN

PRINT 'Departament: ' + @DepartmentName;

SELECT @msg = STRING\_AGG('Employee ID: ' + CAST(ue.employee\_id AS NVARCHAR(10)) + ', Name: ' + ue.first\_name + ' ' + ue.last\_name, CHAR(13) + CHAR(10))

FROM #UpdatedEmployees ue

JOIN employees e ON ue.employee\_id = e.employee\_id

JOIN departments d ON e.department\_id = d.department\_id

WHERE d.department\_name = @DepartmentName;

PRINT @msg;

END

ELSE

BEGIN

PRINT 'Brak pracowników na stanowisku ' + @CurrentJobTitle + ' w departamencie ' + @DepartmentName + ' w kraju ' + @Country + '!';

END;

FETCH NEXT FROM departmentCursor INTO @DepartmentName;

END;

CLOSE departmentCursor;

DEALLOCATE departmentCursor;

DROP TABLE #UpdatedEmployees;

END

GO

DECLARE @UpdatedCount INT;

EXEC changeJobTitle

@CurrentJobTitle = 'Sales Manager',

@NewJobTitle = 'Sales Representative',

@Country = 'United Kingdom',

@UpdatedCount = @UpdatedCount OUTPUT;

PRINT 'Liczba zmodyfikowanych rekordów: ' + CAST(@UpdatedCount AS NVARCHAR(10));

