Bazy Danych - Laboratorium 8

Student:

```
Bartłomiej Krawczyk
```

Numer Albumu:

```
310774
```

```
-- 1. Napisz prosty blok anonimowy zawierający blok wykonawczy z instrukcją NULL.
Uruchom ten program.

BEGIN

    NULL;

END;
/
-- 2. Zmodyfikuj program powyżej i wykorzystaj procedurę dbms_output.put_line
przyjmującą jako parametr łańcuch znakowy do wyświetlenia na konsoli. Uruchom
program i odnajdź napis.


BEGIN

    DBMS_OUTPUT.PUT_LINE( 'Hello, World!');

END;
/

-- 3. Napisz blok anonimowy który doda do tabeli region nowy rekord (np.
‘Oceania’). Uruchom program i zweryfikuj działanie.


BEGIN

    INSERT INTO regions (name) VALUES
    ( 'Oceania');

END;
/

SELECT * FROM regions;
```

```
DELETE FROM regions
WHERE name = 'Oceania';


-- 4. Napisz blok anonimowy, który wygeneruje błąd (RAISE_APPLICATION_ERROR
przyjmującą 2 parametry: kod błędu oraz wiadomość)


BEGIN

    RAISE_APPLICATION_ERROR(-20000, 'Error!');

EXCEPTION

WHEN OTHERS THEN
    NULL;

END;
/

--------------------------------------------------------------------------------

DECLARE
    v_id NUMBER := 102;
    v_name VARCHAR2 (50);
    v_surname employees.surname%TYPE;
    v_employee employees%ROWTYPE;
    c_magic CONSTANT NUMBER := 10;
BEGIN
    -- dbms_output.put_line( 'Employee with id ' || v_id || ' has name ' || v_name
|| ' ' || v_surname);
    SELECT name, surname
    INTO v_name, v_surname
    FROM employees
    WHERE employee_id = v_id;
    -- dbms_output.put_line( ' Employee with id ' || v_id|| ' has name ' || v_name
|| ' ' || v_surname);
    v_id := v_id + length(v_surname) + c_magic;
    SELECT *
    INTO v_employee
    FROM employees
    WHERE employee_id = v_id;
    -- dbms_output.put_line( ' Employee with id ' || v_id || ' has name '||
v_employee.name || ' '|| v_employee.surname);
    INSERT INTO countries(country_id,name,capital) VALUES
(129,'Islandia','Reykjavík');
END;
/

ROLLBACK;

DELETE FROM countries
WHERE country_id = 129;
COMMIT;
```

```
--------------------------------------------------------------------------------

-- 1. Napisz blok anonimowy który będzie korzystał z dwóch zmiennych (v_min_sal
oraz v_emp_id) i który będzie wypisywał na ekran imię i nazwisko pracownika
-- o wskazanym id tylko jeśli jego zarobki są wyższe niż v_min_sal.

DECLARE
    v_min_sal NUMBER := 100;
    v_emp_id NUMBER := 101;
    v_employee employees%ROWTYPE;
BEGIN

    SELECT *
    INTO v_employee
    FROM employees
    WHERE employee_id = v_emp_id;

    IF v_employee.salary > v_min_sal THEN
        -- dbms_output.put_line( 'Employee with id '|| v_emp_id|| ' has name '||
v_employee.name || ' '|| v_employee.surname);
    END IF;

END;
/

--------------------------------------------------------------------------------

CREATE OR replace FUNCTION calculate_seniority_bonus(p_id NUMBER)
RETURN NUMBER
AS
    v_age NUMBER;
    v_yrs_employed NUMBER;
    v_birth_date DATE;
    v_date_employed DATE;
    v_salary NUMBER;
    v_bonus NUMBER := 0;
    c_sal_multiplier CONSTANT NUMBER := 2;
    c_age_min CONSTANT NUMBER := 30;
    c_emp_min CONSTANT NUMBER := 3;
BEGIN
    SELECT birth_date,date_employed, salary
    INTO v_birth_date, v_date_employed, v_salary
    FROM employees
    WHERE employee_id = p_id;
    v_age := extract (year FROM SYSDATE) - extract (year FROM v_birth_date);
    v_yrs_employed := extract (year FROM SYSDATE) - extract (year FROM
v_date_employed);
    IF v_age > c_age_min AND v_yrs_employed > c_emp_min THEN
    v_bonus := c_sal_multiplier * v_salary;
    END IF;
    RETURN v_bonus;
END;
/
```

```
--------------------------------------------------------------------------------
-- Wylicz dodatek stażowy dla pracownika 104.
SELECT calculate_seniority_bonus(104) FROM dual;
/
-- Wylicz dodatek stażowy dla wszystkich pracowników
SELECT e.*, calculate_seniority_bonus (employee_id)
FROM employees e;
/
-- Pokaż maksymalne dodatki stażowe w departamentach. Pokaż liczbę pracowników
departamentu.
SELECT d.name, count (employee_id)AS liczba,
nvl(to_char(max(calculate_seniority_bonus(employee_id))), 'BRAK BONUSU') AS
max_bonus
FROM employees e right join departments d USING (department_id)
GROUP BY d.name
ORDER BY 2 DESC;
/
-- Pokaż wysokości dodatków i liczbę pracowników, którzy go otrzymali. Wyłącz
kandydatów i emerytów.
SELECT calculate_seniority_bonus(employee_id), count(*)
FROM employees e join emp_status s USING (status_id)
WHERE s.name NOT IN ( 'Kandydat', 'Emeryt' )
GROUP BY calculate_seniority_bonus(employee_id)
ORDER BY 1 DESC;
/

--------------------------------------------------------------------------------

-- 1. Napisz funkcję, która wyliczy roczną wartość podatku pracownika. Zakładamy
podatek progresywny. Początkowo stawka to 15%, po przekroczeniu progu 100000
stawka wynosi 25%.

CREATE OR REPLACE FUNCTION year_tax(p_id NUMBER)
RETURN NUMBER
AS
    v_tax                   NUMBER;
    v_salary                NUMBER;
    v_year_salary           NUMBER;
    v_max_first_tax         NUMBER;
    c_first_tax             CONSTANT NUMBER := 0.15;
    c_second_tax            CONSTANT NUMBER := 0.25;
    c_max_value_first_tax   CONSTANT NUMBER := 100000;
    c_months                CONSTANT NUMBER := 12;
BEGIN
    SELECT salary
    INTO v_salary
    FROM employees
    WHERE employee_id = p_id;

    v_year_salary := c_months * v_salary;
    v_max_first_tax := c_max_value_first_tax * c_first_tax;

    IF v_year_salary <= c_max_value_first_tax THEN
        v_tax := v_year_salary * c_first_tax;
```

```
    ELSE
        v_tax := v_max_first_tax + (v_year_salary - c_max_value_first_tax) *
c_second_tax;
    END IF;

    RETURN v_tax;
END;
/

SELECT e.*, year_tax(e.employee_id)
FROM employees e;

-- 2. Stwórz widok łączący departamenty, adresy i kraje. Napisz zapytanie, które
pokaże sumę zapłaconych podatków w krajach.

CREATE OR REPLACE VIEW departments_addresses_countries AS
    SELECT d.department_id, c.country_id, c.name
    FROM departments d
        JOIN addresses a ON d.address_id = a.address_id
        JOIN countries c ON a.country_id = c.country_id;

SELECT d.name, SUM(YEAR_TAX(e.employee_id))
FROM departments_addresses_countries d
JOIN employees e USING (department_id)
GROUP BY d.name;

-- 3. Napisz funkcję, która wyliczy dodatek funkcyjny dla kierowników zespołów.
Dodatek funkcyjny powinien wynosić 10% pensji za każdego podległego pracownika,
ale nie może przekraczać 50% miesięcznej pensji.

CREATE OR REPLACE FUNCTION functional_bonus(p_id NUMBER)
RETURN NUMBER
AS
    v_salary                NUMBER;
    v_subordinate           NUMBER;
    v_bonus                 NUMBER;
    c_functional_bonus      CONSTANT NUMBER := 0.1;
    c_functional_bonus_cap  CONSTANT NUMBER := 5;
BEGIN
    SELECT salary, (SELECT COUNT(*) FROM employees e WHERE e.manager_id = p_id)
subordinates
    INTO v_salary, v_subordinate
    FROM employees man
    WHERE employee_id = p_id;

    IF v_subordinate > c_functional_bonus_cap THEN
        v_subordinate := c_functional_bonus_cap;
    END IF;

    v_bonus := v_subordinate * v_salary * c_functional_bonus;

    RETURN v_bonus;
END;
/
```

```
SELECT e.employee_id, e.salary, FUNCTIONAL_BONUS(e.employee_id) FROM employees e;

-- 4. Zmodyfikuj funkcję calculate_total_bonus, żeby wyliczała całość dodatku dla
pracownika (stażowy i funkcyjny).

CREATE OR REPLACE FUNCTION calculate_total_bonus(p_id NUMBER)
RETURN NUMBER
AS
    v_bonus                 NUMBER;
BEGIN
    v_bonus := calculate_seniority_bonus(p_id) + functional_bonus(p_id);

    RETURN v_bonus;
END;
/

SELECT e.employee_id, e.salary, CALCULATE_TOTAL_BONUS(e.employee_id) FROM
employees e;

--------------------------------------------------------------------------------

CREATE OR REPLACE PROCEDURE add_candidate (p_name VARCHAR2, p_surname VARCHAR2,
p_birth_date DATE, p_gender VARCHAR2, p_pos_name VARCHAR2, p_dep_name VARCHAR2)
AS
    v_pos_id NUMBER;
    v_dep_id NUMBER;
    v_cand_num NUMBER;
    c_candidate_status CONSTANT NUMBER := 304;
    c_num_max CONSTANT NUMBER := 2;
BEGIN
    SELECT position_id INTO v_pos_id FROM positions WHERE name = p_pos_name;
    SELECT department_id INTO v_dep_id FROM departments WHERE name = p_dep_name;
    SELECT count(employee_id) INTO v_cand_num
    FROM employees
    WHERE department_id = v_dep_id AND status_id = c_candidate_status;
    IF v_cand_num < c_num_max THEN
        INSERT INTO employees
        VALUES (NULL, p_name, p_surname, p_birth_date, p_gender,
c_candidate_status, NULL, NULL, v_dep_id, v_pos_id, NULL);
        dbms_output.put_line ( 'Dodano kandydata '|| p_name|| ' '|| p_surname);
    ELSE
        dbms_output.put_line ( 'Za duzo kandydatów w departamencie: '||
p_dep_name);
    END IF;
EXCEPTION
WHEN no_data_found THEN
    dbms_output.put_line ( 'Niepoprawna nazwa stanowiska i/lub zakładu');
    RAISE;
WHEN too_many_rows THEN
    dbms_output.put_line ( 'Nieunikalna nazwa stanowiska i/lub zakładu');
    RAISE;
END;
/
```

```
--------------------------------------------------------------------------------

-- 1. Napisz procedurę, która wykona zmianę stanowiska pracownika. Procedura
powinna przyjmować identyfikator pracownika oraz identyfikator jego nowego
stanowiska.

CREATE OR REPLACE PROCEDURE change_position (p_id NUMBER, p_position_id NUMBER)
AS
BEGIN
    UPDATE employees
    SET
        position_id = p_position_id
    WHERE
        employee_id = p_id;
END;
/

EXEC CHANGE_POSITION(101, 101);

-- 2. Sprawdź działanie procedury wywołując ją z bloku anonimowego.

SELECT employee_id, position_id FROM employees;

BEGIN
    CHANGE_POSITION(101, 102);
END;
/

SELECT employee_id, position_id FROM employees;

ROLLBACK;

-- 3. Napisz procedurę, która zdegraduje zespołowego kierownika o danym
identyfikatorze. Na nowego kierownika zespołu powołaj najstarszego z jego
dotychczasowych podwładnych.

CREATE OR REPLACE PROCEDURE degrade_manager(p_id NUMBER)
AS
    v_new_manager_id NUMBER;
BEGIN
    SELECT employee_id
    INTO v_new_manager_id
    FROM employees
    WHERE
        manager_id = 101
    ORDER BY birth_date
    FETCH FIRST 1 ROW ONLY;

    UPDATE employees
    SET
        manager_id = NULL
    WHERE
        employee_id = v_new_manager_id;
```

```
    UPDATE employees
    SET
        manager_id = v_new_manager_id
    WHERE
        manager_id = p_id
        OR
        employee_id = p_id;


EXCEPTION
WHEN no_data_found THEN
    dbms_output.put_line ( 'Pracownik o podanym id nie istnieje / nie ma
podwładnych');
    RAISE;
END;
/


-- 4. Sprawdź działanie procedury.

SELECT * FROM employees WHERE manager_id = 101 OR employee_id = 101;

EXEC DEGRADE_MANAGER(101);

SELECT * FROM employees WHERE manager_id = 102 OR employee_id = 102;

ROLLBACK;

--------------------------------------------------------------------------------

-- 1. Napisz funkcję, która będzie tworzyła bazowy login dla każdego pracownika.
Login ma się składać z pierwszej litery imienia i maksymalnie 7 znaków z nazwiska.

CREATE OR REPLACE FUNCTION generate_login(p_id NUMBER)
RETURN VARCHAR2
AS
    v_login VARCHAR2(8 CHAR);
BEGIN
    SELECT SUBSTR(name, 1, 1) || SUBSTR(surname, 1, 7)
    INTO v_login
    FROM employees
    WHERE employee_id = p_id;

    RETURN v_login;
END;
/

CREATE OR REPLACE FUNCTION generate_login_from_name(p_name VARCHAR2, p_surname
VARCHAR2)
RETURN VARCHAR2
AS
    v_login VARCHAR2(8 CHAR);
BEGIN
```

```
    v_login := SUBSTR(p_name, 1, 1) || SUBSTR(p_surname, 1, 7);

    RETURN v_login;
END;
/

SELECT name, surname, GENERATE_LOGIN(employee_id)
FROM employees;

SELECT name, surname, generate_login_from_name(name, surname)
FROM employees;

-- 2. Napisz procedurę, która będzie zapisywać login pracownika do nowej kolumny w
tabeli employees (dodaj ją). Zadbaj o to, żeby zapisywany login był unikalny (np.
poprzez dodanie numerów do bazowego loginu).

ALTER TABLE employees
ADD (
    login VARCHAR2(40 BYTE) UNIQUE
);


CREATE OR REPLACE PROCEDURE add_logins(p_id NUMBER)
AS
    v_name VARCHAR2(40 BYTE);
    v_surname VARCHAR2(40 BYTE);
BEGIN
    SELECT name, surname
    INTO v_name, v_surname
    FROM employees
    WHERE employee_id = p_id;

    UPDATE employees
    SET login = generate_login_from_name(v_name, v_surname) || p_id
    WHERE employee_id = p_id;
END;
/

SELECT * FROM employees;
EXEC add_logins(101);
SELECT * FROM employees;

-- 3. Sprawdź działanie trybów przekazania parametrów do procedury (IN, IN OUT i
OUT).

CREATE OR REPLACE PROCEDURE test_procedure(
        p_in IN NUMBER,
        p_in_out IN OUT NUMBER,
        p_out OUT NUMBER
    )
AS
BEGIN
    DBMS_OUTPUT.PUT_LINE(p_in);
    DBMS_OUTPUT.PUT_LINE(p_in_out);
```

```
    DBMS_OUTPUT.PUT_LINE(p_out);
--    p_in := 4;
    p_in_out := 5;
    p_out := 6;
END;
/
DECLARE
    a NUMBER := 1;
    b NUMBER := 2;
    c NUMBER := 3;
BEGIN

    DBMS_OUTPUT.PUT_LINE(a);
    DBMS_OUTPUT.PUT_LINE(b);
    DBMS_OUTPUT.PUT_LINE(c);
    DBMS_OUTPUT.PUT_LINE( '---');
    TEST_PROCEDURE(a, b, c);
    DBMS_OUTPUT.PUT_LINE( '---');
    DBMS_OUTPUT.PUT_LINE(a);
    DBMS_OUTPUT.PUT_LINE(b);
    DBMS_OUTPUT.PUT_LINE(c);

END;
/
```