```
SELECT * FROM JOBS;


CREATE OR REPLACE PROCEDURE pobierz_nazwe_pracy (
    p_job_id jobs.job_id%TYPE,
    p_job_title OUT jobs.job_title%TYPE
) AS
    BEGIN
        SELECT job_title INTO p_job_title FROM JOBS
            WHERE job_id = p_job_id;
        DBMS_OUTPUT.PUT_LINE('Stanowisko: ' || p_job_title);
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('Brak stanowiska o ID: ' || p_job_id);
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Błąd: ' || SQLERRM);
    END;
/


SET SERVEROUTPUT ON;


DECLARE
    v_tytul jobs.job_title%TYPE;
BEGIN
    pobierz_nazwe_pracy('IT_PROG', v_tytul);
END;
/
```

```
SELECT * FROM EMPLOYEES;


CREATE OR REPLACE PROCEDURE pobierz_roczne_zarobki (
    p_emp_id employees.employee_id%TYPE,
    p_roczna_kwota OUT NUMBER
) AS
    v_salary employees.salary%TYPE;
    v_commission_pct employees.commission_pct%TYPE;
    BEGIN
        SELECT salary, commission_pct INTO v_salary, v_commission_pct FROM EMPLOYEES
            WHERE employee_id = p_emp_id;
        IF v_commission_pct IS NULL THEN
            v_commission_pct := 0;
        END IF;

        p_roczna_kwota := v_salary * 12 + (v_salary * v_commission_pct);


        DBMS_OUTPUT.PUT_LINE('Roczne zarobki: ' || p_roczna_kwota);
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('Nie znaleziono pracownika o ID: ' || p_emp_id);
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Błąd: ' || SQLERRM);
    END;
/


DECLARE
```

```
    v_kwota NUMBER;
BEGIN
    pobierz_roczne_zarobki(149, v_kwota);
END;
/

CREATE OR REPLACE PROCEDURE wyciagnij_kierunkowy_po_nazwisku (
    p_last_name employees.last_name%TYPE,
    p_kierunkowy OUT VARCHAR2
) AS
    v_telefon employees.phone_number%TYPE;
    v_pozycja_kropki PLS_INTEGER;
    BEGIN
        SELECT phone_number INTO v_telefon FROM EMPLOYEES
        WHERE LOWER(last_name) = LOWER(p_last_name) AND ROWNUM = 1;
    v_pozycja_kropki := INSTR(v_telefon, '.');
    IF v_pozycja_kropki > 0 THEN
        p_kierunkowy := SUBSTR(v_telefon, 1, v_pozycja_kropki - 1);
    ELSE
        p_kierunkowy := v_telefon;
    END IF;
    DBMS_OUTPUT.PUT_LINE('Kierunkowy dla ' || p_last_name || ': ' || p_kierunkowy);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nie znaleziono pracownika o nazwisku: ' || p_last_name);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Błąd: ' || SQLERRM);
```

```
    END;
/

DECLARE
    v_kierunkowy VARCHAR2(20);
BEGIN
    wyciagnij_kierunkowy_po_nazwisku('King', v_kierunkowy);
END;
/

CREATE OR REPLACE PROCEDURE formatuj_napis (
    p_wejscie IN  VARCHAR2,
    p_wyjscie OUT VARCHAR2
) AS
    v_dlugosc PLS_INTEGER;
BEGIN
    v_dlugosc := LENGTH(p_wejscie);
    IF v_dlugosc = 0 THEN
        p_wyjscie := '';
    ELSIF v_dlugosc = 1 THEN
        p_wyjscie := UPPER(p_wejscie);
    ELSE
        p_wyjscie :=
            UPPER(SUBSTR(p_wejscie, 1, 1)) ||
            LOWER(SUBSTR(p_wejscie, 2, v_dlugosc - 2)) ||
            UPPER(SUBSTR(p_wejscie, -1, 1));
    END IF;
```

```
        DBMS_OUTPUT.PUT_LINE('Sformatowany ciąg: ' || p_wyjscie);
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Błąd: ' || SQLERRM);
    END;
/

DECLARE
    v_wynik VARCHAR2(100);
BEGIN
    formatuj_napis('wHaTsSuP', v_wynik);
END;
/

CREATE OR REPLACE PROCEDURE pesel_na_date (
    p_pesel IN  VARCHAR2,
    p_data OUT DATE
) AS
    v_rok NUMBER;
    v_mies NUMBER;
    v_dzien NUMBER;
    BEGIN
        v_rok  := TO_NUMBER(SUBSTR(p_pesel, 1, 2));
        v_mies := TO_NUMBER(SUBSTR(p_pesel, 3, 2));
        v_dzien := TO_NUMBER(SUBSTR(p_pesel, 5, 2));
        IF v_mies > 20 THEN
            v_rok := 2000 + v_rok;
```

```
        v_mies := v_mies - 20;
    ELSE
        v_rok := 1900 + v_rok;
    END IF;
    p_data := TO_DATE(v_rok || '-' || LPAD(v_mies, 2, '0') || '-' || LPAD(v_dzien, 2, '0'), 'YYYY-MM-DD');
    DBMS_OUTPUT.PUT_LINE('Data urodzenia: ' || TO_CHAR(p_data, 'YYYY-MM-DD'));
  EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Błąd: ' || SQLERRM);
  END;
/

DECLARE
  v_data DATE;
BEGIN
  pesel_na_date('02270812345', v_data);
END;
/

SELECT * FROM COUNTRIES;
SELECT * FROM DEPARTMENTS;

CREATE OR REPLACE PROCEDURE statystyki_kraju (
  p_country_name countries.country_name%TYPE,
  p_liczba_pracownikow OUT NUMBER,
  p_liczba_departamentow OUT NUMBER
) AS
```

```
v_country_id COUNTRIES.country_id%TYPE;
BEGIN
    SELECT country_id INTO v_country_id FROM COUNTRIES
        WHERE LOWER(country_name) = LOWER(p_country_name);
    SELECT COUNT(*) INTO p_liczba_departamentow FROM DEPARTMENTS d JOIN locations l ON d.location_id = l.location_id
        WHERE l.country_id = v_country_id;
    SELECT COUNT(*) INTO p_liczba_pracownikow FROM EMPLOYEES e JOIN departments d ON e.department_id = d.department_id JOIN locations l ON d.location_id = l.location_id
        WHERE l.country_id = v_country_id;
    DBMS_OUTPUT.PUT_LINE('Liczba departamentów: ' || p_liczba_departamentow);
    DBMS_OUTPUT.PUT_LINE('Liczba pracowników: ' || p_liczba_pracownikow);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nie znaleziono kraju o nazwie: ' || p_country_name);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Błąd: ' || SQLERRM);
    END;
/

DECLARE
    v_prac NUMBER;
    v_dep  NUMBER;
BEGIN
    statystyki_kraju('Egypt', v_prac, v_dep);
END;
/

CREATE TABLE ARCHIWUM_DEPARTAMENTOW (
```

```
    id NUMBER,
    nazwa VARCHAR2(100),
    data_zamkniecia DATE,
    ostatni_manager VARCHAR2(100)
);

SELECT * FROM ARCHIWUM_DEPARTAMENTOW;

CREATE OR REPLACE TRIGGER trgr_archiwum_departamentow
    AFTER DELETE ON departments
        FOR EACH ROW DECLARE
            v_manager_name VARCHAR2(100);
    BEGIN
        SELECT first_name || ' ' || last_name INTO v_manager_name FROM EMPLOYEES
        WHERE employee_id = :OLD.manager_id;
    INSERT INTO archiwum_departamentow(id, nazwa, data_zamkniecia, ostatni_manager) VALUES
        (
        :OLD.department_id,
        :OLD.department_name,
        SYSDATE,
        v_manager_name
        );
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        INSERT INTO ARCHIWUM_DEPARTAMENTOW(id, nazwa, data_zamkniecia, ostatni_manager) VALUES
        (
        :OLD.department_id,
```

```
            :OLD.department_name,
            SYSDATE,
            'Nieznany'
            );
    END;
/

ALTER TABLE EMPLOYEES DISABLE CONSTRAINT department_id_in_employees;
ALTER TABLE JOB_HISTORY DISABLE CONSTRAINT department_id_in_job_history;

DELETE FROM DEPARTMENTS
    WHERE department_id = 50;

SELECT * FROM ARCHIWUM_DEPARTAMENTOW;
SELECT * FROM DEPARTMENTS;

CREATE TABLE ZLODZIEJ (
    id NUMBER GENERATED ALWAYS AS IDENTITY,
    "USER" VARCHAR2(30),
    czas_zmiany DATE
);

CREATE OR REPLACE TRIGGER trgr_sala_range_guard
    BEFORE INSERT OR UPDATE ON EMPLOYEES
        FOR EACH ROW
        BEGIN
            IF :NEW.salary < 2000 OR :NEW.salary > 26000 THEN
```

```
            INSERT INTO ZLODZIEJ("USER", czas_zmiany)
            VALUES (USER, SYSDATE);
            --RAISE_APPLICATION_ERROR(-20001, 'Wynagrodzenie poza dozwolonym zakresem (2000–26000)');
        END IF;
    END;
/

INSERT INTO EMPLOYEES (
    employee_id,
    first_name,
    last_name,
    email,
    phone_number,
    hire_date,
    job_id,
    salary,
    manager_id,
    department_id
)
VALUES (
    666,
    'Zly',
    'Czlowiek',
    'zly@test.com',
    '233.334.124',
    SYSDATE,
    'IT_PROG',
```

```
    30000,
    100,
    60
);

SELECT * FROM ZLODZIEJ;
SELECT * FROM EMPLOYEES;

CREATE SEQUENCE seq_employees_id START WITH 1000 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER trgr_emp_auto_id
    BEFORE
        INSERT ON EMPLOYEES
            FOR EACH ROW
            BEGIN
                IF :NEW.employee_id IS NULL THEN
                    SELECT seq_employees_id.NEXTVAL INTO :NEW.employee_id FROM dual;
                END IF;
            END;
/

INSERT INTO EMPLOYEES
(
    first_name,
    last_name,
    email,
    job_id,
```

```
    hire_date,
    salary,
    department_id
)
VALUES
(
    'Auto1',
    '1',
    'autoid1@test.com',
    'IT_PROG',
    SYSDATE,
    4000,
    60
);

SELECT * FROM EMPLOYEES;


CREATE OR REPLACE TRIGGER trgr_block_job_grades
    BEFORE
        INSERT OR UPDATE OR DELETE ON JOB_GRADES
            BEGIN
                RAISE_APPLICATION_ERROR(-20002, 'Zmiany w tabeli JOB_GRADES są zabronione');
            END;
    /


DELETE FROM JOB_GRADES WHERE grade = 'A';
```

```
SELECT * FROM JOB_GRADES;

ALTER TRIGGER trgr_block_job_grades DISABLE;

DELETE FROM JOB_GRADES WHERE grade_level = 'A';

ALTER TRIGGER trgr_block_job_grades ENABLE;

SELECT trigger_name, status FROM user_triggers
   WHERE table_name = 'JOB_GRADES';

CREATE OR REPLACE TRIGGER trg_jobs_block_salary_change
   BEFORE UPDATE ON JOBS
      FOR EACH ROW
         BEGIN
            :NEW.min_salary := :OLD.min_salary;
            :NEW.max_salary := :OLD.max_salary;
         END;
/

UPDATE JOBS SET min_salary = 9999, max_salary = 99999
   WHERE job_id = 'IT_PROG';

ALTER TRIGGER trg_jobs_block_salary_change DISABLE;

UPDATE JOBS SET min_salary = 9999, max_salary = 99999
   WHERE job_id = 'IT_PROG';
```

```
ALTER TRIGGER trg_jobs_block_salary_change ENABLE;

SELECT trigger_name, status FROM user_triggers
    WHERE table_name = 'JOBS';

SELECT * FROM JOBS;

CREATE OR REPLACE PACKAGE moja_paczka AS
    PROCEDURE dodaj_job(p_job_id jobs.job_id%TYPE, p_job_title jobs.job_title%TYPE);
    PROCEDURE dodaj_pracownika(
        p_first_name employees.first_name%TYPE,
        p_last_name employees.last_name%TYPE,
        p_email employees.email%TYPE,
        p_salary employees.salary%TYPE,
        p_job_id employees.job_id%TYPE,
        p_hire_date DATE DEFAULT SYSDATE,
        p_department_id employees.department_id%TYPE
    );
    PROCEDURE pobierz_nazwe_pracy(p_job_id jobs.job_id%TYPE, p_job_title OUT jobs.job_title%TYPE);
    PROCEDURE pesel_na_date(p_pesel IN VARCHAR2, p_data OUT DATE);
END moja_paczka;
/

CREATE OR REPLACE PACKAGE regions_pck AS
    PROCEDURE dodaj_region(p_id NUMBER, p_nazwa VARCHAR2);
    PROCEDURE usun_region(p_id NUMBER);
```

```
    PROCEDURE aktualizuj_region(p_id NUMBER, p_nowa_nazwa VARCHAR2);
    PROCEDURE pokaz_region_po_id(p_id NUMBER);
    PROCEDURE pokaz_region_po_nazwie(p_fragment VARCHAR2);
    PROCEDURE pokaz_wszystkie;
END regions_pck;
/
```