**PAKIET 1 – ZARZĄDZANIE ZAMÓWIENIAMI**

**FUNKCJE**

1. **GENERUJ FAKTURE SPRZEDAZY –** funkcja generująca informacje potrzebne do faktury sprzedaży, zawierająca numer faktury, nazwy zabawek znajdujących się w zamówieniu oraz łączną kwotę tego zamówienia.

CREATE OR REPLACE FUNCTION GenerujFaktureSprzedazy(nr\_faktury\_p IN INTEGER)

RETURN VARCHAR2

IS

v\_numer\_faktury INTEGER;

v\_nazwy\_zabawek VARCHAR2(4000);

v\_kwota NUMBER := 0;

CURSOR c\_pozycje\_zamowienia IS

SELECT Z.nazwa, P.ilosc\_sztuk, Z.cena

FROM Pozycje\_Zamowienia P

JOIN Zabawki Z ON P.id\_zabawki = Z.id\_zabawki

WHERE P.id\_zamowienia = v\_numer\_faktury;

BEGIN

SELECT id\_zamowienia INTO v\_numer\_faktury

FROM Faktury\_sprzedazy

WHERE nr\_faktury = nr\_faktury\_p;

v\_nazwy\_zabawek := '';

FOR rec IN c\_pozycje\_zamowienia LOOP

v\_nazwy\_zabawek := v\_nazwy\_zabawek || rec.nazwa || ', ';

v\_kwota := v\_kwota + (rec.cena \* rec.ilosc\_sztuk);

END LOOP;

RETURN 'Numer faktury: ' || nr\_faktury\_p || CHR(10) ||

'Zabawki: ' || v\_nazwy\_zabawek || CHR(10) ||

'Kwota zamówienia: ' || TO\_CHAR(v\_kwota, '99999.99');

END;

/

DECLARE

v\_wynik VARCHAR2(4000);

v\_numer\_faktury INTEGER := 1;

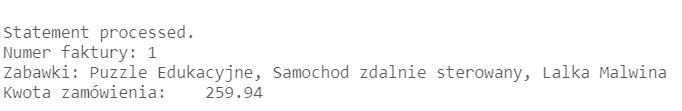
BEGIN

v\_wynik := GenerujFaktureSprzedazy(v\_numer\_faktury);

DBMS\_OUTPUT.PUT\_LINE(v\_wynik);

END;

/



1. **RAPORT ZAMÓWIEŃ Z BIEŻĄCEGO ROKU –** funkcja przedstawiająca raport, który jest zestawieniem zamówień dokonanych przez klientów w bieżącym roku, zawierający nazwę klienta, datę zamówienia, nazwę zabawki oraz jej cenę.

CREATE OR REPLACE FUNCTION Raport\_Zamowien\_Biezacy\_Rok RETURN INTEGER IS

v\_ilosc INTEGER := 0;

CURSOR zamowienia\_cursor IS

SELECT K.nazwa\_firmy, Z.data\_zamowienia,

D.nazwa AS nazwa\_zabawki, D.cena

FROM Zamowienia Z

JOIN Klienci K ON Z.id\_klienta = K.id\_klienta

JOIN Pozycje\_Zamowienia PZ ON Z.id\_zamowienia = PZ.id\_zamowienia

JOIN Zabawki D ON PZ.id\_zabawki = D.id\_zabawki

WHERE EXTRACT(YEAR FROM Z.data\_zamowienia) = EXTRACT(YEAR FROM SYSDATE);

BEGIN

BEGIN

FOR rekord IN zamowienia\_cursor LOOP

v\_ilosc := v\_ilosc + 1;

DBMS\_OUTPUT.PUT\_LINE('Nazwa Firmy: ' || rekord.nazwa\_firmy);

DBMS\_OUTPUT.PUT\_LINE('Data Zamówienia: ' || rekord.data\_zamowienia);

DBMS\_OUTPUT.PUT\_LINE('Zamówiona zabawka: ' || rekord.nazwa\_zabawki);

DBMS\_OUTPUT.PUT\_LINE('Cena: ' || rekord.cena);

DBMS\_OUTPUT.PUT\_LINE('---------------------------------------');

END LOOP;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Brak danych dla bieżącego roku.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Wystąpił błąd: ' || SQLERRM);

END;

DBMS\_OUTPUT.PUT\_LINE('Ilość zamówień z bieżącego roku: ' || v\_ilosc);

RETURN v\_ilosc;

END Raport\_Zamowien\_Biezacy\_Rok;

/



DECLARE

ilosc\_zamowien INTEGER;

BEGIN

ilosc\_zamowien := Raport\_Zamowien\_Biezacy\_Rok;

END;

/

1. **TOP5 najczęściej zamawianych produktów –** funkcja tworzy zestawienie pięciu najczęściej sprzedawanych zabawek, przedstawia ilość i nazwę

CREATE OR REPLACE FUNCTION Raport\_Najlepiej\_Sprzedawane\_Zabawki RETURN INTEGER IS

v\_ilosc INTEGER := 0;

CURSOR najlepiej\_sprzedawane\_cursor IS

SELECT D.id\_zabawki, D.nazwa, SUM(PZ.ilosc\_sztuk) AS ilosc\_sprzedanych

FROM Pozycje\_Zamowienia PZ

JOIN Zamowienia Z ON PZ.id\_zamowienia = Z.id\_zamowienia

JOIN Zabawki D ON PZ.id\_zabawki = D.id\_zabawki

GROUP BY D.id\_zabawki, D.nazwa

ORDER BY ilosc\_sprzedanych DESC;

BEGIN

BEGIN

FOR rekord IN najlepiej\_sprzedawane\_cursor LOOP

v\_ilosc := v\_ilosc + 1;

DBMS\_OUTPUT.PUT\_LINE('Miejsce ' || v\_ilosc || ':');

DBMS\_OUTPUT.PUT\_LINE('ID Zabawki: ' || rekord.id\_zabawki);

DBMS\_OUTPUT.PUT\_LINE('Nazwa Zabawki: ' || rekord.nazwa);

DBMS\_OUTPUT.PUT\_LINE('Ilość Sprzedanych: ' || rekord.ilosc\_sprzedanych);

DBMS\_OUTPUT.PUT\_LINE('---------------------------------------');

IF v\_ilosc = 5 THEN

EXIT;

END IF;

END LOOP;

EXCEPTION

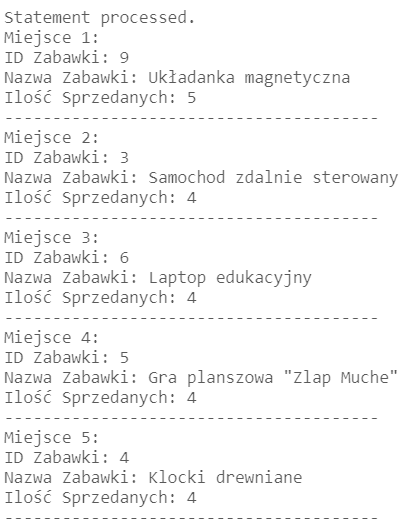
WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Brak danych do wyświetlenia.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Wystąpił błąd: ' || SQLERRM);

END;



RETURN v\_ilosc;

END Raport\_Najlepiej\_Sprzedawane\_Zabawki;

/

DECLARE

ilosc INTEGER;

BEGIN

ilosc := Raport\_Najlepiej\_Sprzedawane\_Zabawki;

DBMS\_OUTPUT.PUT\_LINE('Liczba najlepiej sprzedawanych zabawek: ' || ilosc);

END;

/

1. **Ilość zamówień klienta –** funkcja zliczająca ilość zamówień wykonanych przez danego klienta, podawanego poprzez numer id

CREATE OR REPLACE FUNCTION Pobierz\_Ilosc\_Zamowien\_Klienta(p\_id\_klienta INTEGER) RETURN INTEGER IS

ilosc\_zamowien INTEGER;

v\_nazwa VARCHAR2(20);

CURSOR klient\_cursor IS

SELECT nazwa\_firmy

FROM Klienci

WHERE id\_klienta = p\_id\_klienta;

CURSOR zamowienia\_cursor IS

SELECT COUNT(\*)

FROM Zamowienia

WHERE id\_klienta = p\_id\_klienta;

BEGIN

OPEN klient\_cursor;

FETCH klient\_cursor INTO v\_nazwa;

CLOSE klient\_cursor;

DBMS\_OUTPUT.PUT\_LINE('Klient o ID ' || p\_id\_klienta || ': ' || v\_nazwa );

OPEN zamowienia\_cursor;

FETCH zamowienia\_cursor INTO ilosc\_zamowien;

CLOSE zamowienia\_cursor;

DBMS\_OUTPUT.PUT\_LINE('Ilość zamówień: ' || ilosc\_zamowien);

RETURN ilosc\_zamowien;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Brak danych do wyświetlenia.');

RETURN 0;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Wystąpił nieoczekiwany błąd: ' || SQLERRM);

RETURN 0;

END Pobierz\_Ilosc\_Zamowien\_Klienta;

/

DECLARE

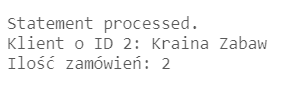
v\_ilosc\_zamowien INTEGER;

BEGIN

v\_ilosc\_zamowien := Pobierz\_Ilosc\_Zamowien\_Klienta(2);

END;

/



**PROCEDURY**

1. **Realizuj zamówienie –** procedura mająca na celu dodanie wiersza do tabeli zamówienia

CREATE OR REPLACE PROCEDURE RealizujZamowienie(

p\_id\_klienta INTEGER,

p\_data\_zamowienia DATE,

p\_produkty\_tab SYS.ODCINUMBERLIST,

p\_id\_sposobu\_zaplaty INTEGER

) AS

v\_id\_zamowienia INTEGER;

v\_id\_pozycji\_zamowienia INTEGER;

BEGIN

SELECT COALESCE(MAX(id\_zamowienia) + 1, 1)

INTO v\_id\_zamowienia

FROM Zamowienia;

SELECT COALESCE(MAX(id\_pozycji\_zamowienia) + 1, 1)

INTO v\_id\_pozycji\_zamowienia

FROM Pozycje\_Zamowienia;

INSERT INTO Zamowienia (id\_zamowienia, id\_klienta, data\_zamowienia, status\_zamowienia, id\_sposobu\_zaplaty)

VALUES (v\_id\_zamowienia, p\_id\_klienta, p\_data\_zamowienia, 'W trakcie', p\_id\_sposobu\_zaplaty);

FOR i IN 1..p\_produkty\_tab.COUNT LOOP

INSERT INTO Pozycje\_Zamowienia (id\_pozycji\_zamowienia, id\_zamowienia, id\_zabawki, ilosc\_sztuk)

VALUES (v\_id\_pozycji\_zamowienia, v\_id\_zamowienia, p\_produkty\_tab(i), 1);

v\_id\_pozycji\_zamowienia := v\_id\_pozycji\_zamowienia + 1;

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Zamówienie o ID ' || v\_id\_zamowienia || ' zostało zrealizowane.');

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Wystąpił błąd podczas realizacji zamówienia: ' || SQLERRM);

END RealizujZamowienie;

/

DECLARE

v\_produkty SYS.ODCINUMBERLIST := SYS.ODCINUMBERLIST(7, 2, 11, 4);

BEGIN

RealizujZamowienie(

p\_id\_klienta => 1,

p\_data\_zamowienia => SYSDATE,

p\_produkty\_tab => v\_produkty,

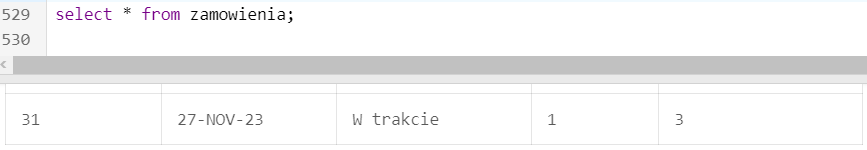
p\_id\_sposobu\_zaplaty => 3

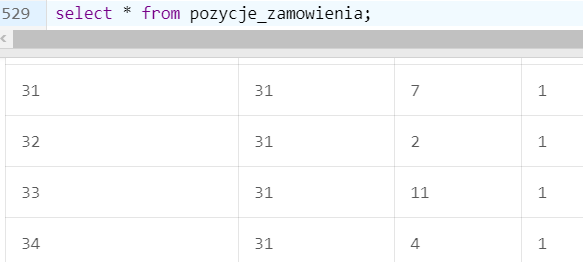
);

END;

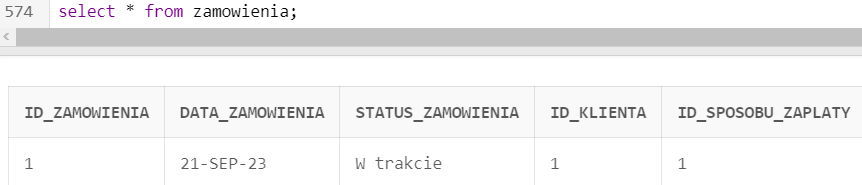
/







1. **Zmiana statusu zamówienia –** procedura aktualizująca komórkę odpowiedzialną za status zamówienia w tabeli zamówienia



CREATE OR REPLACE PROCEDURE ZmienStatusZamowienia(

p\_id\_zamowienia INTEGER,

p\_nowy\_status VARCHAR2

) AS

CURSOR zamowienie\_cursor IS

SELECT id\_zamowienia, status\_zamowienia

FROM Zamowienia

WHERE id\_zamowienia = p\_id\_zamowienia;

v\_id\_zamowienia INTEGER;

v\_stary\_status VARCHAR2(50);

BEGIN

OPEN zamowienie\_cursor;

FETCH zamowienie\_cursor INTO v\_id\_zamowienia, v\_stary\_status;

CLOSE zamowienie\_cursor;

IF v\_id\_zamowienia IS NOT NULL THEN

UPDATE Zamowienia

SET status\_zamowienia = p\_nowy\_status

WHERE id\_zamowienia = v\_id\_zamowienia;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Zmieniono status zamówienia o ID ' || v\_id\_zamowienia || ' z ' || v\_stary\_status || ' na ' || p\_nowy\_status);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Nie znaleziono zamówienia o ID ' || p\_id\_zamowienia);

END IF;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Wystąpił błąd podczas zmiany statusu zamówienia: ' || SQLERRM);

END ZmienStatusZamowienia;

/

DECLARE

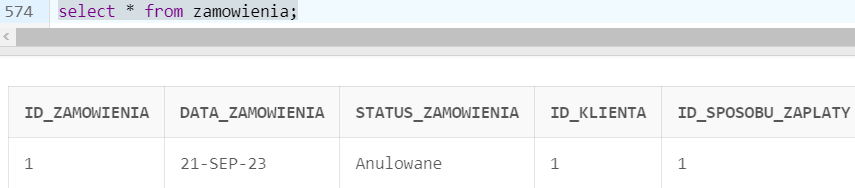
v\_id\_zamowienia INTEGER := 1;

v\_nowy\_status VARCHAR2(50) := 'Anulowane';

BEGIN

ZmienStatusZamowienia(v\_id\_zamowienia, v\_nowy\_status);

END;



1. **Zmiana danych klientów –** aktualizacja danych zawartych w wierszu w tabeli Klienci

CREATE OR REPLACE PROCEDURE ZmienDaneKlienta(

p\_id\_klienta INTEGER,

p\_nowa\_nazwa\_firmy VARCHAR2,

p\_nowe\_imie VARCHAR2,

p\_nowe\_nazwisko VARCHAR2,

p\_nowy\_pesel VARCHAR2,

p\_nowe\_id\_adresu INTEGER,

p\_nowy\_nr\_ulicy VARCHAR2

) AS

CURSOR klient\_cursor IS

SELECT id\_klienta, nazwa\_firmy, imie, nazwisko, pesel, id\_adresu, nr\_ulicy

FROM Klienci

WHERE id\_klienta = p\_id\_klienta;

v\_id\_klienta INTEGER;

v\_stara\_nazwa\_firmy VARCHAR2(100);

v\_stare\_imie VARCHAR2(50);

v\_stare\_nazwisko VARCHAR2(50);

v\_stary\_pesel VARCHAR2(11);

v\_stare\_id\_adresu INTEGER;

v\_stary\_nr\_ulicy VARCHAR2(20);

BEGIN

OPEN klient\_cursor;

FETCH klient\_cursor INTO v\_id\_klienta, v\_stara\_nazwa\_firmy, v\_stare\_imie, v\_stare\_nazwisko, v\_stary\_pesel, v\_stare\_id\_adresu, v\_stary\_nr\_ulicy;

CLOSE klient\_cursor;

IF v\_id\_klienta IS NOT NULL THEN

BEGIN

SELECT id\_klienta

INTO v\_id\_klienta

FROM Klienci

WHERE id\_klienta = p\_id\_klienta;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

NULL;

END;

UPDATE Klienci

SET nazwa\_firmy = p\_nowa\_nazwa\_firmy,

imie = p\_nowe\_imie,

nazwisko = p\_nowe\_nazwisko,

pesel = p\_nowy\_pesel,

id\_adresu = p\_nowe\_id\_adresu,

nr\_ulicy = p\_nowy\_nr\_ulicy

WHERE id\_klienta = p\_id\_klienta;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Zmieniono dane klienta o ID ' || p\_id\_klienta);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Nie znaleziono klienta o ID ' || p\_id\_klienta);

END IF;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Wystąpił błąd podczas zmiany danych klienta: ' || SQLERRM);

END ZmienDaneKlienta;

/

DECLARE

v\_id\_klienta INTEGER := 1;

v\_nowa\_nazwa\_firmy VARCHAR2(100) := 'Kanał zabawkowy';

v\_nowe\_imie VARCHAR2(50) := 'Michał';

v\_nowe\_nazwisko VARCHAR2(50) := 'Pol';

v\_nowy\_pesel VARCHAR2(11) := '463547372';

v\_nowe\_id\_adresu INTEGER := 14;

v\_nowy\_nr\_ulicy VARCHAR2(20) := '5A';

BEGIN

ZmienDaneKlienta(

p\_id\_klienta => v\_id\_klienta,

p\_nowa\_nazwa\_firmy => v\_nowa\_nazwa\_firmy,

p\_nowe\_imie => v\_nowe\_imie,

p\_nowe\_nazwisko => v\_nowe\_nazwisko,

p\_nowy\_pesel => v\_nowy\_pesel,

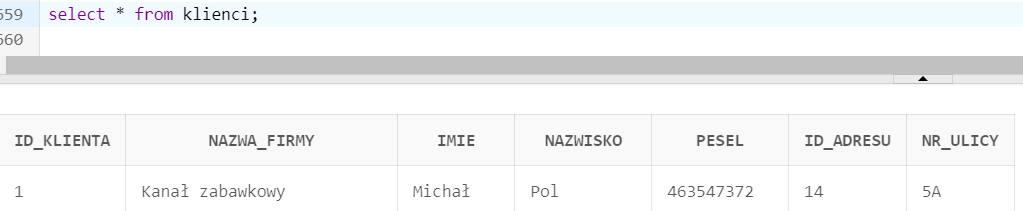
p\_nowe\_id\_adresu => v\_nowe\_id\_adresu,

p\_nowy\_nr\_ulicy => v\_nowy\_nr\_ulicy

);

END;

/



1. **DODAWANIE FAKTURY SPRZEDAZY –** PROCEDURA MAJĄCA NA CELU DODANIE WIERSZA DO TABELI FAKTURY\_SPRZEDAZY

CREATE OR REPLACE PROCEDURE Dodaj\_Fakture\_Sprzedazy (

p\_id\_zamowienia INTEGER

) AS

v\_nr\_faktury INTEGER;

v\_data\_wystawienia DATE := SYSDATE;

BEGIN

SELECT COALESCE(MAX(nr\_faktury) + 1, 1)

INTO v\_nr\_faktury

FROM Faktury\_sprzedazy;

INSERT INTO Faktury\_sprzedazy (nr\_faktury, id\_zamowienia, data\_wystawienia)

VALUES (v\_nr\_faktury, p\_id\_zamowienia, v\_data\_wystawienia);

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Faktura sprzedaży o numerze ' || v\_nr\_faktury || ' została dodana.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Brak danych dla zamówienia o ID ' || p\_id\_zamowienia);

ROLLBACK;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Błąd: ' || SQLERRM);

ROLLBACK;

END Dodaj\_Fakture\_Sprzedazy;

/

DECLARE

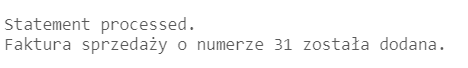
v\_id\_zamowienia INTEGER := 6;

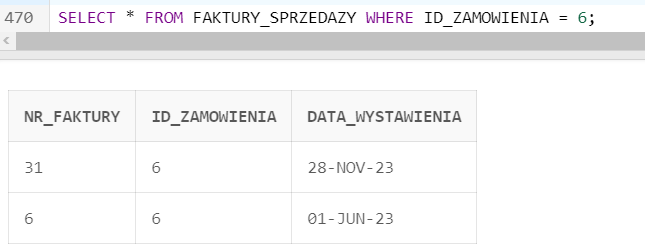
BEGIN

Dodaj\_Fakture\_Sprzedazy(v\_id\_zamowienia);

END;

/





**PAKIET 2 – ZARZĄDZANIE MAGAZYNEM**

**PROCEDURY**

1. **Zmien dane pracowników –** procedura pozwalająca na edytowanie danych dotyczących pracowników w tabeli Pracownicy

CREATE OR REPLACE PROCEDURE Zmien\_Informacje\_O\_Pracowniku (

p\_id\_pracownika INTEGER,

p\_nowe\_imie VARCHAR2,

p\_nowe\_nazwisko VARCHAR2,

p\_nowy\_nr\_telefonu INTEGER,

p\_nowe\_wynagrodzenie INTEGER,

p\_nowe\_stanowisko VARCHAR2,

p\_nowy\_id\_adresu INTEGER,

p\_nowy\_nr\_ulicy VARCHAR2

)

IS

CURSOR pracownik\_cursor IS

SELECT \*

FROM Pracownicy

WHERE id\_pracownika = p\_id\_pracownika;

v\_id\_pracownika INTEGER;

v\_stare\_imie VARCHAR2(50);

v\_stare\_nazwisko VARCHAR2(50);

v\_stary\_nr\_telefonu INTEGER;

v\_stare\_wynagrodzenie INTEGER;

v\_stare\_stanowisko VARCHAR2(50);

v\_stare\_id\_adresu INTEGER;

v\_stary\_nr\_ulicy VARCHAR2(20);

BEGIN

OPEN pracownik\_cursor;

FETCH pracownik\_cursor INTO v\_id\_pracownika, v\_stare\_imie, v\_stare\_nazwisko, v\_stary\_nr\_telefonu, v\_stare\_wynagrodzenie, v\_stare\_stanowisko, v\_stare\_id\_adresu, v\_stary\_nr\_ulicy;

CLOSE pracownik\_cursor;

IF v\_id\_pracownika IS NOT NULL THEN

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Zmiana danych pracownika o ID ' || p\_id\_pracownika || ':');

DBMS\_OUTPUT.PUT\_LINE('Stare dane:');

DBMS\_OUTPUT.PUT\_LINE('Imię: ' || v\_stare\_imie);

DBMS\_OUTPUT.PUT\_LINE('Nazwisko: ' || v\_stare\_nazwisko);

DBMS\_OUTPUT.PUT\_LINE('Nr telefonu: ' || v\_stary\_nr\_telefonu);

DBMS\_OUTPUT.PUT\_LINE('Wynagrodzenie: ' || v\_stare\_wynagrodzenie);

DBMS\_OUTPUT.PUT\_LINE('Stanowisko: ' || v\_stare\_stanowisko);

DBMS\_OUTPUT.PUT\_LINE('ID adresu: ' || v\_stare\_id\_adresu);

DBMS\_OUTPUT.PUT\_LINE('Nr ulicy: ' || v\_stary\_nr\_ulicy);

UPDATE Pracownicy

SET

imie = p\_nowe\_imie,

nazwisko = p\_nowe\_nazwisko,

nr\_telefonu = p\_nowy\_nr\_telefonu,

wynagrodzenie\_podstawowe = p\_nowe\_wynagrodzenie,

stanowisko = p\_nowe\_stanowisko,

id\_adresu = p\_nowy\_id\_adresu,

nr\_ulicy = p\_nowy\_nr\_ulicy

WHERE id\_pracownika = p\_id\_pracownika;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Nowe dane:');

DBMS\_OUTPUT.PUT\_LINE('Imię: ' || p\_nowe\_imie);

DBMS\_OUTPUT.PUT\_LINE('Nazwisko: ' || p\_nowe\_nazwisko);

DBMS\_OUTPUT.PUT\_LINE('Nr telefonu: ' || p\_nowy\_nr\_telefonu);

DBMS\_OUTPUT.PUT\_LINE('Wynagrodzenie: ' || p\_nowe\_wynagrodzenie);

DBMS\_OUTPUT.PUT\_LINE('Stanowisko: ' || p\_nowe\_stanowisko);

DBMS\_OUTPUT.PUT\_LINE('ID adresu: ' || p\_nowy\_id\_adresu);

DBMS\_OUTPUT.PUT\_LINE('Nr ulicy: ' || p\_nowy\_nr\_ulicy);

DBMS\_OUTPUT.PUT\_LINE('Informacje o pracowniku zaktualizowane pomyślnie.');

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Błąd: ' || SQLERRM);

END;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Nie znaleziono pracownika o ID ' || p\_id\_pracownika);

END IF;

END Zmien\_Informacje\_O\_Pracowniku;

/

DECLARE

v\_id\_pracownika INTEGER := 13;

v\_nowe\_imie VARCHAR2(50) := 'Adrian';

v\_nowe\_nazwisko VARCHAR2(50) := 'Jakubowski';

v\_nowy\_nr\_telefonu INTEGER := 788788788;

v\_nowe\_wynagrodzenie INTEGER := 5000;

v\_nowe\_stanowisko VARCHAR2(50) := 'Magazynier';

v\_nowy\_id\_adresu INTEGER := 10;

v\_nowy\_nr\_ulicy VARCHAR2(20) := '20';

BEGIN

Zmien\_Informacje\_O\_Pracowniku(

p\_id\_pracownika => v\_id\_pracownika,

p\_nowe\_imie => v\_nowe\_imie,

p\_nowe\_nazwisko => v\_nowe\_nazwisko,

p\_nowy\_nr\_telefonu => v\_nowy\_nr\_telefonu,

p\_nowe\_wynagrodzenie => v\_nowe\_wynagrodzenie,

p\_nowe\_stanowisko => v\_nowe\_stanowisko,

p\_nowy\_id\_adresu => v\_nowy\_id\_adresu,

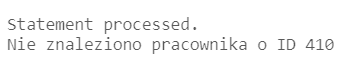
p\_nowy\_nr\_ulicy => v\_nowy\_nr\_ulicy

);

END;

/



****

1. **Dodanie wpisu magazynowego –** procedura polegająca na dodaniu wiersza do tabeli wpisy\_magazynowe

CREATE OR REPLACE PROCEDURE Dodaj\_Wpis\_Magazynowy (

p\_data\_wpisu DATE,

p\_ilosc\_sztuk INTEGER,

p\_id\_zabawki INTEGER,

p\_id\_pracownika INTEGER

) AS

v\_id\_wpisu INTEGER;

CURSOR id\_wpisu\_cursor IS

SELECT COALESCE(MAX(id\_wpisu) + 1, 1) AS next\_id

FROM Wpisy\_magazynowe;

v\_next\_id INTEGER;

BEGIN

OPEN id\_wpisu\_cursor;

FETCH id\_wpisu\_cursor INTO v\_next\_id;

CLOSE id\_wpisu\_cursor;

INSERT INTO Wpisy\_Magazynowe (id\_wpisu, data\_wpisu, ilosc\_sztuk, id\_zabawki, id\_pracownika)

VALUES (v\_next\_id, p\_data\_wpisu, p\_ilosc\_sztuk, p\_id\_zabawki, p\_id\_pracownika);

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Wpis magazynowy został dodany');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Brak danych do wstawienia.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Błąd: ' || SQLERRM);

ROLLBACK;

END Dodaj\_Wpis\_Magazynowy;

/

DECLARE

v\_data\_wpisu DATE := SYSDATE;

v\_ilosc\_sztuk INTEGER := 10;

v\_id\_zabawki INTEGER := 1;

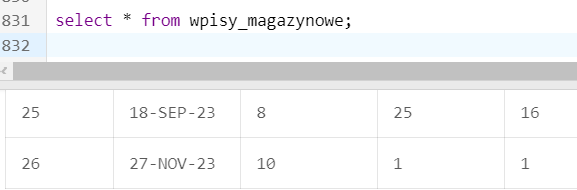
v\_id\_pracownika INTEGER := 1;

BEGIN

Dodaj\_Wpis\_Magazynowy(v\_data\_wpisu, v\_ilosc\_sztuk, v\_id\_zabawki, v\_id\_pracownika);

END;

/



1. **Zaktualizuj liczbe zabawek – procedura aktualizująca komórkę odpowiedzialną za dostępną liczbę zabawek w tabeli Zabawki**

CREATE OR REPLACE PROCEDURE Aktualizuj\_Ilosc\_Zabawek (

p\_id\_zabawki INTEGER,

p\_nowa\_ilosc INTEGER

)

IS

CURSOR zabawki\_c IS

SELECT dostepna\_ilosc

FROM Zabawki

WHERE id\_zabawki = p\_id\_zabawki;

v\_stara\_ilosc INTEGER;

BEGIN

OPEN zabawki\_c;

FETCH zabawki\_c INTO v\_stara\_ilosc;

IF v\_dostepna\_ilosc IS NOT NULL THEN

UPDATE Zabawki

SET dostepna\_ilosc = p\_nowa\_ilosc

WHERE id\_zabawki = p\_id\_zabawki;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Ilość zabawek została zaktualizowana.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Zabawka o ID ' || p\_id\_zabawki || ' nie znaleziona');

END IF;

CLOSE zabawki\_c;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Bład: ' || SQLERRM);

ROLLBACK;

END Aktualizuj\_Ilosc\_Zabawek;

/

DECLARE

v\_id\_zabawki INTEGER := 5;

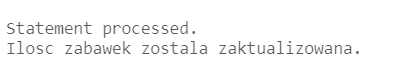
v\_nowa\_ilosc INTEGER := 47;

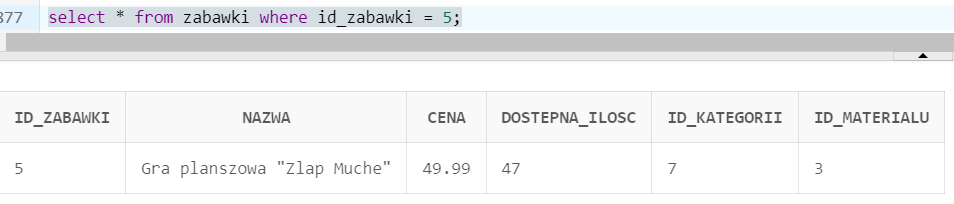
BEGIN

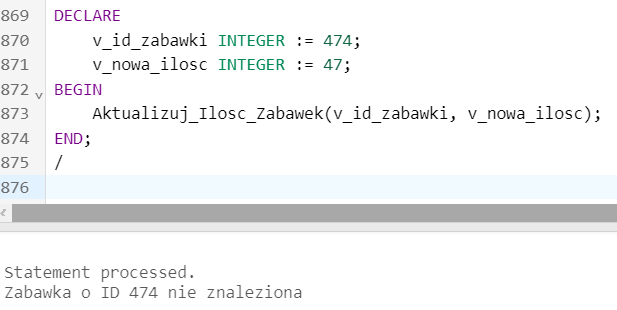
Aktualizuj\_Ilosc\_Zabawek(v\_id\_zabawki, v\_nowa\_ilosc);

END;

/







1. **Dodawanie zabawki –** procedura dodająca wiersz do tabeli Zabawki

CREATE OR REPLACE PROCEDURE Dodaj\_Zabawke (

p\_nazwa VARCHAR2,

p\_cena NUMBER,

p\_dostepna\_ilosc INTEGER,

p\_id\_kategorii INTEGER,

p\_id\_materialu INTEGER

)

AS

v\_id\_zabawki INTEGER;

CURSOR nowa\_zabawka\_cursor IS

SELECT \*

FROM Zabawki

WHERE id\_zabawki = v\_id\_zabawki;

v\_rekord Zabawki%ROWTYPE;

BEGIN

SELECT COALESCE(MAX(id\_zabawki) + 1, 1)

INTO v\_id\_zabawki

FROM Zabawki;

INSERT INTO Zabawki (id\_zabawki, nazwa, cena, dostepna\_ilosc, id\_kategorii, id\_materialu)

VALUES (v\_id\_zabawki, p\_nazwa, p\_cena, p\_dostepna\_ilosc, p\_id\_kategorii, p\_id\_materialu);

COMMIT;

OPEN nowa\_zabawka\_cursor;

FETCH nowa\_zabawka\_cursor INTO v\_rekord;

CLOSE nowa\_zabawka\_cursor;

DBMS\_OUTPUT.PUT\_LINE('Zabawka została dodana:');

DBMS\_OUTPUT.PUT\_LINE('ID: ' || v\_rekord.id\_zabawki);

DBMS\_OUTPUT.PUT\_LINE('Nazwa: ' || v\_rekord.nazwa);

DBMS\_OUTPUT.PUT\_LINE('Cena: ' || v\_rekord.cena);

DBMS\_OUTPUT.PUT\_LINE('Dostepna ilosc: ' || v\_rekord.dostepna\_ilosc);

DBMS\_OUTPUT.PUT\_LINE('ID kategorii: ' || v\_rekord.id\_kategorii);

DBMS\_OUTPUT.PUT\_LINE('ID materialu: ' || v\_rekord.id\_materialu);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Brak danych do wstawienia.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Błąd: ' || SQLERRM);

ROLLBACK;

END Dodaj\_Zabawke;

/

DECLARE

v\_nazwa VARCHAR2(50) := 'Drewniane Puzzle';

v\_cena NUMBER := 19.99;

v\_dostepna\_ilosc INTEGER := 50;

v\_id\_kategorii INTEGER := 10;

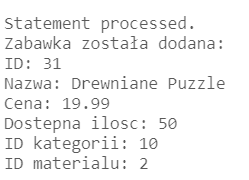
v\_id\_materialu INTEGER := 2;

BEGIN

Dodaj\_Zabawke(v\_nazwa, v\_cena, v\_dostepna\_ilosc, v\_id\_kategorii, v\_id\_materialu);

END;

/



**FUNKCJE**

1. **Srednia pensja na stanowisku –** funkcja obliczająca średnią pensję na danym stanowisku.

CREATE OR REPLACE FUNCTION Oblicz\_Srednia\_Pensje\_Pracownikow\_Na\_Stanowisku(

p\_stanowisko VARCHAR2

)

RETURN NUMBER

IS

v\_srednia\_pensja NUMBER;

CURSOR pensje\_cursor IS

SELECT wynagrodzenie\_podstawowe

FROM Pracownicy

WHERE stanowisko = p\_stanowisko;

BEGIN

OPEN pensje\_cursor;

FETCH pensje\_cursor INTO v\_srednia\_pensja;

IF v\_srednia\_pensja IS NULL THEN

CLOSE pensje\_cursor;

DBMS\_OUTPUT.PUT\_LINE('Brak danych dla stanowiska: ' || p\_stanowisko);

RETURN NULL;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Pobrano dane: ' || v\_srednia\_pensja);

END IF;

CLOSE pensje\_cursor;

RETURN v\_srednia\_pensja;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Błąd: ' || SQLERRM);

RETURN NULL;

END Oblicz\_Srednia\_Pensje\_Pracownikow\_Na\_Stanowisku;

/

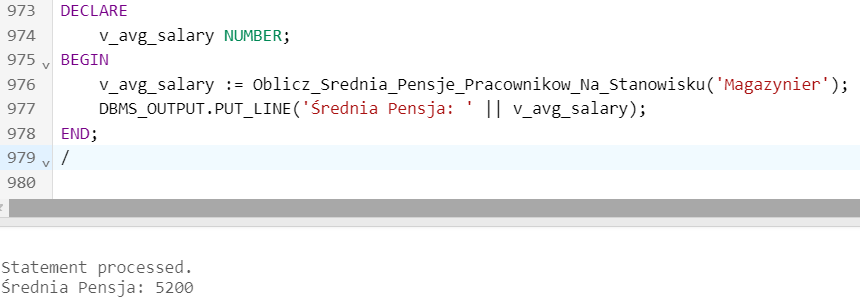
DECLARE

v\_avg\_salary NUMBER;

BEGIN

v\_avg\_salary := Oblicz\_Srednia\_Pensje\_Pracownikow\_Na\_Stanowisku('Magazynier');

DBMS\_OUTPUT.PUT\_LINE('Średnia Pensja: ' || v\_avg\_salary);

END;

/

1. **Raport wpisów za bieżący miesiąc –** funkcja generująca raport wpisów magazynowych w bieżącym miesiącu

CREATE OR REPLACE FUNCTION Raport\_Wpisow\_Magazynowych\_Biezacy\_Miesiac RETURN INTEGER IS

v\_ilosc INTEGER := 0;

CURSOR wpisy\_cursor IS

SELECT W.data\_wpisu, P.imie || ' ' || P.nazwisko AS pracownik,

Z.nazwa AS nazwa\_zabawki, W.ilosc\_sztuk

FROM Wpisy\_Magazynowe W

JOIN Pracownicy P ON W.id\_pracownika = P.id\_pracownika

JOIN Zabawki Z ON W.id\_zabawki = Z.id\_zabawki

WHERE EXTRACT(MONTH FROM W.data\_wpisu) = EXTRACT(MONTH FROM SYSDATE)

AND EXTRACT(YEAR FROM W.data\_wpisu) = EXTRACT(YEAR FROM SYSDATE);

BEGIN

FOR rekord IN wpisy\_cursor LOOP

v\_ilosc := v\_ilosc + 1;

DBMS\_OUTPUT.PUT\_LINE('Data wpisu: ' || rekord.data\_wpisu);

DBMS\_OUTPUT.PUT\_LINE('Pracownik: ' || rekord.pracownik);

DBMS\_OUTPUT.PUT\_LINE('Zabawka: ' || rekord.nazwa\_zabawki);

DBMS\_OUTPUT.PUT\_LINE('Ilość sztuk: ' || rekord.ilosc\_sztuk);

DBMS\_OUTPUT.PUT\_LINE('---------------------------------------');

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Ilość wpisów magazynowych z bieżącego miesiąca: ' || v\_ilosc);

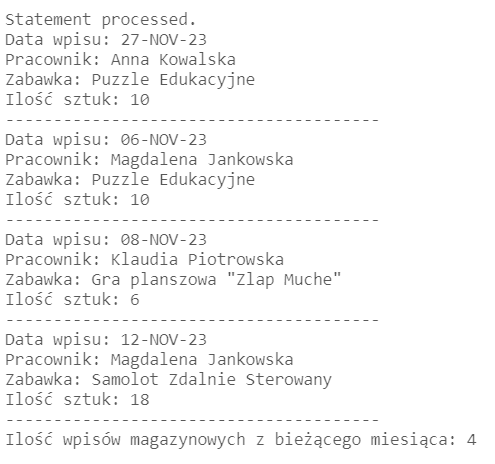
RETURN v\_ilosc;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Brak danych dla bieżącego miesiąca.');

RETURN 0;

**** WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Błąd: ' || SQLERRM);

RETURN NULL;

END Raport\_Wpisow\_Magazynowych\_Biezacy\_Miesiac;

/

DECLARE

raport\_wpisow INTEGER;

BEGIN

raport\_wpisow := Raport\_Wpisow\_Magazynowych\_Biezacy\_Miesiac;

END;

/

1. **Liczba stworzonych zabawek w okresie czasu –** funkcja wyliczająca ilość stworzonych zabawek w danym okresie czasu

CREATE OR REPLACE FUNCTION Ilosc\_Zabawek\_W\_Okresie(

p\_data\_poczatkowa DATE,

p\_data\_koncowa DATE

) RETURN INTEGER IS

v\_ilosc INTEGER := 0;

CURSOR zabawki\_cursor IS

SELECT COUNT(\*) AS liczba\_zabawek

FROM Wpisy\_magazynowe

WHERE data\_wpisu BETWEEN p\_data\_poczatkowa AND p\_data\_koncowa;

BEGIN

OPEN zabawki\_cursor;

FETCH zabawki\_cursor INTO v\_ilosc;

CLOSE zabawki\_cursor;

IF v\_ilosc > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Ilość zabawek stworzonych w okresie od ' || p\_data\_poczatkowa || ' do ' || p\_data\_koncowa || ': ' || v\_ilosc);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Brak zabawek w podanym okresie.');

END IF;

RETURN v\_ilosc;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Wystąpił błąd: ' || SQLERRM);

END Ilosc\_Zabawek\_W\_Okresie;

/

DECLARE

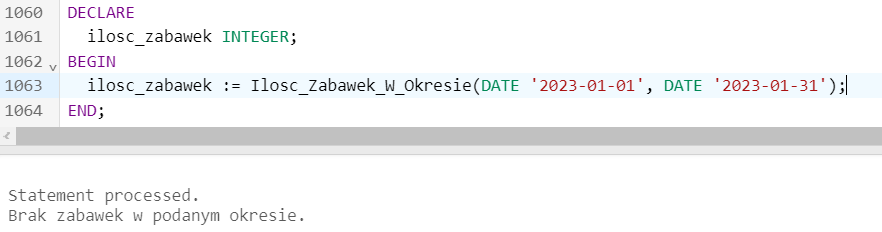
ilosc\_zabawek INTEGER;

BEGIN

ilosc\_zabawek := Ilosc\_Zabawek\_W\_Okresie(DATE '2023-04-01', DATE '2023-05-31');

END;





1. **Srednia cena zabawek w danej kategorii –** funkcja obliczająca średnią cenę zabawek dla zadanej kategorii

CREATE OR REPLACE FUNCTION Srednia\_Cena\_Zabawki\_W\_Kategorii(

p\_id\_kategorii INTEGER

) RETURN NUMBER IS

v\_srednia\_cena NUMBER;

CURSOR cena\_cursor IS

SELECT AVG(cena) AS srednia\_cena

FROM Zabawki

WHERE id\_kategorii = p\_id\_kategorii;

BEGIN

OPEN cena\_cursor;

FETCH cena\_cursor INTO v\_srednia\_cena;

CLOSE cena\_cursor;

IF v\_srednia\_cena IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Średnia cena zabawki w kategorii o ID ' || p\_id\_kategorii || ': ' || v\_srednia\_cena);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Brak zabawek w kategorii o ID ' || p\_id\_kategorii);

END IF;

RETURN v\_srednia\_cena;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Wystąpił błąd: ' || SQLERRM);

RETURN NULL;

END Srednia\_Cena\_Zabawki\_W\_Kategorii;

/

DECLARE

v\_id\_kategorii INTEGER := 1;

v\_srednia\_cena NUMBER;

BEGIN

v\_srednia\_cena := Srednia\_Cena\_Zabawki\_W\_Kategorii(v\_id\_kategorii);

END;

/

