

# PAKIET 1 – ZARZĄDZANIE ZAMÓWIENIAMI

## FUNKCJE

- 1. GENERUJ FAKTURE SPRZEDAŻY** – 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;
/
```

```
Statement processed.
Numer faktury: 1
Zabawki: Puzzle Edukacyjne, Samochod zdalnie sterowany, Lalka Malwina
Kwota zamówienia:      259.94
```

**2. 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;
/
```

```
Nazwa Firmy: Magiczne Zabawki
Data Zamówienia: 21-SEP-23
Zamówiona zabawka: Lalka Malwina
Cena: 39.99
-----
Nazwa Firmy: Magiczne Zabawki
Data Zamówienia: 21-SEP-23
Zamówiona zabawka: Samochod zdalnie sterowany
Cena: 79.99
-----
Nazwa Firmy: Bajkowe Zabawki
Data Zamówienia: 01-JUL-23
Zamówiona zabawka: Samochod zdalnie sterowany
Cena: 79.99
-----
Nazwa Firmy: Skarby Dziecinstwa
Data Zamówienia: 01-NOV-23
Zamówiona zabawka: Klocki drewniane
Cena: 29.99
-----
Nazwa Firmy: Kraina Zabaw
Data Zamówienia: 01-FEB-23
Zamówiona zabawka: Gra planszowa "Zlap Muche"
Cena: 49.99
-----
Nazwa Firmy: Skarby Dziecinstwa
Data Zamówienia: 01-NOV-23
Zamówiona zabawka: Laptop edukacyjny
Cena: 89.99
```

### 3. 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;
```

```
/
```

```
Statement processed.
Miejsce 1:
ID Zabawki: 9
Nazwa Zabawki: Układanka magnetyczna
Ilość Sprzedanych: 5
-----
Miejsce 2:
ID Zabawki: 3
Nazwa Zabawki: Samochód zdalnie sterowany
Ilość Sprzedanych: 4
-----
Miejsce 3:
ID Zabawki: 6
Nazwa Zabawki: Laptop edukacyjny
Ilość Sprzedanych: 4
-----
Miejsce 4:
ID Zabawki: 5
Nazwa Zabawki: Gra planszowa "Złap Muche"
Ilość Sprzedanych: 4
-----
Miejsce 5:
ID Zabawki: 4
Nazwa Zabawki: Klocki drewniane
Ilość Sprzedanych: 4
-----
```

#### 4. 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;
/
```

Statement processed.  
Klient o ID 2: Kraina Zabaw  
Ilość zamówień: 2

## 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;
/
```

Statement processed.  
Zamówienie o ID 31 zostało zrealizowane.

529 `select * from zamowienia;`

530

<

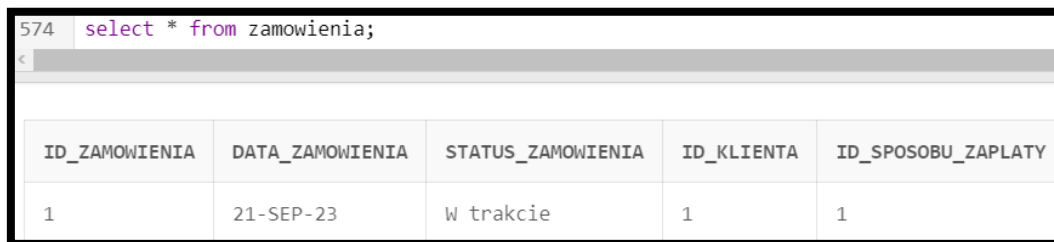
31	27-NOV-23	W trakcie	1	3
----	-----------	-----------	---	---

529 `select * from pozycje_zamowienia;`

<

31	31	7	1
32	31	2	1
33	31	11	1
34	31	4	1

## 2. Zmiana statusu zamówienia - procedura aktualizująca komórkę odpowiedzialną za status zamówienia w tabeli zamówienia



```
574 select * from zamowienia;
```

ID_ZAMOWIENIA	DATA_ZAMOWIENIA	STATUS_ZAMOWIENIA	ID_KLIENTA	ID_SPOSOBU_ZAPLATY
1	21-SEP-23	W trakcie	1	1

```
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;
```

574

```
select * from zamowienia;
```

<

ID_ZAMOWIENIA	DATA_ZAMOWIENIA	STATUS_ZAMOWIENIA	ID_KLIENTA	ID_SPOSOBU_ZAPLATY
1	21-SEP-23	Anulowane	1	1

### 3. 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;
/

```

559 select \* from klienci;

560

ID_KLIENTA	NAZWA_FIRMY	IMIE	NAZWISKO	PESEL	ID_ADRESU	NR_ULICY
1	Kanał zabawkowy	Michał	Pol	463547372	14	5A

#### 4. DODAWANIE FAKTURY SPRZEDAŻY – PROCEDURA MAJĄCA NA CELU DODANIE WIERSCZY DO TABELI FAKTURY\_SPRZEDAŻY

```

CREATE OR REPLACE PROCEDURE Dodaj_Faktura_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;
/

```

Statement processed.  
Faktura sprzedaży o numerze 31 została dodana.

470 SELECT \* FROM FAKTURY\_SPRZEDAZY WHERE ID\_ZAMOWIENIA = 6;

NR_FAKTURY	ID_ZAMOWIENIA	DATA_WYSTAWIENIA
31	6	28-NOV-23
6	6	01-JUN-23

## 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  
            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('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;
/
```

```

642 select * from pracownicy where id_pracownika = 13;
643
```

ID_PRACOWNIKA	IMIE	NAZWISKO	NR_TELEFONU	WYNAGRODZENIE_PODSTAWOWE	STANOWISKO	ID_ADRESU	NR_ULICY
13	Adrian	Jakubowski	788788788	5000	Magazynier	10	20

Statement processed.  
Nie znaleziono pracownika o ID 410

## 2. 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;  
/
```

831

select \* from wpisy\_magazynowe;

832

### 3. 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('Błąd: ' || 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;  
/  

```

Statement processed.  
Ilosc zabawek zostala zaktualizowana.

```
877 select * from zabawki where id_zabawki = 5;
```

ID_ZABAWKI	NAZWA	CENA	DOSTEPNA_ILOSC	ID_KATEGORII	ID_MATERIALU
5	Gra planszowa "Zlap Muche"	49.99	47	7	3

```
869 DECLARE
870     v_id_zabawki INTEGER := 474;
871     v_nowa_ilosc  INTEGER := 47;
872 v BEGIN
873     Aktualizuj_Ilosc_Zabawek(v_id_zabawki, v_nowa_ilosc);
874 END;
875 /
876
```

Statement processed.  
Zabawka o ID 474 nie znaleziona

#### 4. 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;
/
```

```

Statement processed.
Zabawka została dodana:
ID: 31
Nazwa: Drewniane Puzzle
Cena: 19.99
Dostepna ilosc: 50
ID kategorii: 10
ID materialu: 2
```



# FUNKCJE

## 1. Średnia 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;
    END IF;

    LOOP
        FETCH pensje_cursor INTO v_srednia_pensja;
        EXIT WHEN v_srednia_pensja IS NULL;
        DBMS_OUTPUT.PUT_LINE('Pobrano dane: ' || v_srednia_pensja);
    END LOOP;

    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;
/
```

```
973 DECLARE
974     v_avg_salary NUMBER;
975 v BEGIN
976     v_avg_salary := Oblicz_Srednia_Pensje_Pracownikow_Na_Stanowisku('Magazynier');
977     DBMS_OUTPUT.PUT_LINE('Średnia Pensja: ' || v_avg_salary);
978 END;
979 v /
980
```

Statement processed.  
Średnia Pensja: 5200

## 2. 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;
/
```

```
Statement processed.
Data wpisu: 27-NOV-23
Pracownik: Anna Kowalska
Zabawka: Puzzle Edukacyjne
Ilość sztuk: 10
-----
Data wpisu: 06-NOV-23
Pracownik: Magdalena Jankowska
Zabawka: Puzzle Edukacyjne
Ilość sztuk: 10
-----
Data wpisu: 08-NOV-23
Pracownik: Klaudia Piotrowska
Zabawka: Gra planszowa "Zlap Muche"
Ilość sztuk: 6
-----
Data wpisu: 12-NOV-23
Pracownik: Magdalena Jankowska
Zabawka: Samolot Zdalnie Sterowany
Ilość sztuk: 18
-----
Ilość wpisów magazynowych z bieżącego miesiąca: 4
```

### 3. 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;
```

```
Statement processed.
Ilość zabawek stworzonych w okresie od 01-APR-23 do 31-MAY-23: 5
```

```
1060 DECLARE
1061   ilosc_zabawek INTEGER;
1062 v BEGIN
1063   ilosc_zabawek := Ilosc_Zabawek_W_Okresie(DATE '2023-01-01', DATE '2023-01-31');|
1064 END;
```

```
Statement processed.
Brak zabawek w podanym okresie.
```

#### 4. Średnia 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;  
/
```

statement processed.  
Średnia cena zabawki w kategorii o ID 1: 74.99