Laboratorium 11

Implementacja funkcji w języku PL/pgSQL.

Zadanie 11.1

1. Napisz funkcję **masaPudelka** wyznaczającą masę pudełka jako sumę masy czekoladek w nim zawartych. Funkcja jako argument przyjmuje identyfikator pudełka. Przetestuj działanie funkcji na podstawie prostej instrukcji **select**.

```
CREATE OR REPLACE FUNCTION masaPudelka(in arg1 CHARACTER(4))
RETURNS INTEGER AS
$$
DECLARE
        wynik INTEGER;
BEGIN
        SELECT SUM(c.masa*z.sztuk) INTO wynik
        FROM
                pudelka p
                INNER JOIN zawartosc z USING (idpudelka)
                INNER JOIN czekoladki c USING (idczekoladki)
        WHERE p.idpudelka = arg1;

        RETURN wynik;
END;
$$ LANGUAGE PLpgSQL;

SELECT masaPudelka('alls');
```

2. $\star$ Napisz funkcję **liczbaCzekoladek** wyznaczającą liczbę czekoladek znajdujących się w pudełku. Funkcja jako argument przyjmuje identyfikator pudełka. Przetestuj działanie funkcji na podstawie prostej instrukcji **select**.

```
CREATE OR REPLACE FUNCTION liczbaCzekoladek(in arg1 CHARACTER(4))
RETURNS INTEGER AS
$$
DECLARE
        wynik INTEGER;
BEGIN
        SELECT SUM(sztuk) INTO wynik
        FROM zawartosc
        WHERE idpudelka = arg1;

        RETURN wynik;
END;
$$ LANGUAGE PLpgSQL;

SELECT liczbaCzekoladek('alls');
```

Zadanie 11.2

1. Napisz funkcję **zysk** obliczającą zysk jaki cukiernia uzyskuje ze sprzedaży jednego pudełka czekoladek, zakładając, że zysk ten jest różnicą między ceną pudełka, a kosztem wytworzenia zawartych

w nim czekoladek i kosztem opakowania (0,90 zł dla każdego pudełka). Funkcja jako argument przyjmuje identyfikator pudełka. Przetestuj działanie funkcji na podstawie prostej instrukcji **select**.

```
CREATE OR REPLACE FUNCTION zysk(in arg1 CHARACTER(4))
RETURNS NUMERIC(7,2) AS
$$
DECLARE
wynik NUMERIC(7,2);
BEGIN
        SELECT p.cena - SUM(z.sztuk*c.koszt) - 0.9 INTO wynik
        FROM pudelka p
                INNER JOIN zawartosc z USING(idpudelka)
                INNER JOIN czekoladki c USING(idczekoladki)
        WHERE idpudelka = arg1
                GROUP BY idpudelka;

        RETURN wynik;
END;
$$ LANGUAGE PLpgSQL;

SELECT zysk('alls');
```

2. Napisz instrukcję **select** obliczającą zysk jaki cukiernia uzyska ze sprzedaży pudełek zamówionych w wybranym dniu.

```
SELECT SUM(zysk(a.idpudelka)*a.sztuk)
FROM zamowienia z
        INNER JOIN artykuly a USING(idzamowienia)
WHERE z.datarealizacji = '2013-11-12';
```

Zadanie 11.3

1. Napisz funkcję **sumaZamowien** obliczającą łączną wartość zamówień złożonych przez klienta, które czekają na realizację (są w tabeli **Zamowienia**). Funkcja jako argument przyjmuje identyfikator klienta. Przetestuj działanie funkcji.

```
CREATE OR REPLACE FUNCTION sumaZamowien(in arg1 INTEGER)
RETURNS INTEGER AS
$$
DECLARE
wynik INTEGER;
BEGIN
        SELECT SUM(a.sztuk*p.cena) INTO wynik
        FROM
                zamowienia z
                INNER JOIN artykuly a USING(idzamowienia)
                INNER JOIN pudelka p USING(idpudelka)
        WHERE z.idklienta = arg1;

        RETURN wynik;
END;
$$ LANGUAGE PLpgSQL;

SELECT sumaZamowien(1);
```

2. Napisz funkcję **rabat** obliczającą rabat jaki otrzymuje klient składający zamówienie. Funkcja jako argument przyjmuje identyfikator klienta. Rabat wyliczany jest na podstawie wcześniej złożonych zamówień w sposób następujący:

   - ☐ 4 % jeśli wartość zamówień jest z przedziału 101-200 zł;
   - ☐ 7 % jeśli wartość zamówień jest z przedziału 201-400 zł;
   - ☐ 8 % jeśli wartość zamówień jest większa od 400 zł.

```
CREATE OR REPLACE FUNCTION rabat(in arg1 INTEGER)
RETURNS INTEGER AS
$$
DECLARE wynik INTEGER;
DECLARE wartoscZamowien INTEGER;
BEGIN
        SELECT SUM(a.sztuk*p.cena) INTO wartoscZamowien
        FROM
                zamowienia z
                INNER JOIN artykuly a USING(idzamowienia)
                INNER JOIN pudelka p USING(idpudelka)
        WHERE z.idklienta = arg1;

        IF wartoscZamowien BETWEEN 101 AND 200 THEN
                wynik := 4;
        ELSIF wartoscZamowien BETWEEN 201 AND 400 THEN
                wynik := 7;
        ELSIF wartoscZamowien > 400 THEN
                wynik := 8;
        ELSE
                wynik := 0;
        END IF;

        RETURN wynik;
END;
$$ LANGUAGE PLpgSQL;

SELECT rabat(1);
```

Zadanie 11.4

Napisz bezargumentową funkcję **podwyzka**, która dokonuje podwyżki kosztów produkcji czekoladek o:

- ☐ 3 gr dla czekoladek, których koszt produkcji jest mniejszy od 20 gr;
- ☐ 4 gr dla czekoladek, których koszt produkcji jest z przedziału 20-29 gr;
- ☐ 5 gr dla pozostałych.

Funkcja powinna ponadto podnieść cenę pudełek o tyle o ile zmienił się koszt produkcji zawartych w nich czekoladek.

Przed testowaniem działania funkcji wykonaj zapytania, które umieszczą w plikach dane na temat kosztów czekoladek i cen pudełek tak, aby można było później sprawdzić poprawność działania funkcji podwyzka. Przetestuj działanie funkcji.

```
CREATE OR REPLACE FUNCTION podwyzka()
RETURNS void AS
$$
DECLARE zmiana NUMERIC(7,2);
DECLARE c1 RECORD;
DECLARE c2 RECORD;
BEGIN
        FOR c1 IN SELECT * FROM czekoladki LOOP
                zmiana := CASE
                WHEN c1.koszt < 0.20 THEN 0.03
                WHEN c1.koszt BETWEEN 0.20 AND 0.29 THEN 0.04
                ELSE 0.05 END;

                UPDATE czekoladki
                SET koszt = koszt+zmiana
                        WHERE idczekoladki = c1.idczekoladki;

                FOR c2 IN SELECT * FROM zawartosc
                                WHERE idczekoladki = c1.idczekoladki LOOP
                        UPDATE pudelka SET cena = cena + (zmiana*c2.sztuk)
                                WHERE idpudelka = c2.idpudelka;
                END LOOP;
        END LOOP;
END;
$$ LANGUAGE PLpgSQL;

SELECT podwyzka();
```

⋆Zadanie 11.5

Napisz funkcję **obnizka** odwracająca zmiany wprowadzone w poprzedniej funkcji. Przetestuj działanie funkcji.

```
CREATE OR REPLACE FUNCTION obnizka()
RETURNS void AS
$$
DECLARE zmiana NUMERIC(7,2);
DECLARE c1 RECORD;
DECLARE c2 RECORD;
BEGIN
        FOR c1 IN SELECT * FROM czekoladki LOOP
                zmiana := CASE
                WHEN c1.koszt < 0.23 THEN 0.03
                WHEN c1.koszt BETWEEN 0.24 AND 0.33 THEN 0.04
                ELSE 0.05 END;

                UPDATE czekoladki
                SET koszt = koszt-zmiana
                        WHERE idczekoladki = c1.idczekoladki;

                FOR c2 IN SELECT * FROM zawartosc
                                WHERE idczekoladki = c1.idczekoladki LOOP
                        UPDATE pudelka SET cena = cena - (zmiana*c2.sztuk)
                                WHERE idpudelka = c2.idpudelka;
                END LOOP;
```

```
        END LOOP;
END;
$$ LANGUAGE PLpgSQL;

SELECT obnizka();
```

Zadanie 11.6

1. Napisz funkcję zwracającą informacje o zamówieniach złożonych przez klienta, którego identyfikator podawany jest jako argument wywołania funkcji. W/w informacje muszą zawierać: **idzamowienia**, **idpudelka**, **datarealizacji**. Przetestuj działanie funkcji. **Uwaga**: Funkcja zwraca więcej niż 1 wiersz!

```
CREATE TEMPORARY TABLE  IF NOT EXISTS zam(
zamowienie INTEGER,
pudelka    CHAR(4),
data       DATE
);

CREATE OR REPLACE FUNCTION info(in arg1 INTEGER)
RETURNS SETOF zam AS
$$
BEGIN
RETURN QUERY SELECT
                idzamowienia,
                idpudelka,
                datarealizacji
        FROM zamowienia
                INNER JOIN artykuly USING(idzamowienia)
        WHERE idklienta = arg1;
END;
$$
LANGUAGE PLpgSQL;

SELECT info(1);
```

2. ⋆Napisz funkcję zwracającą listę klientów z miejscowości, której nazwa podawana jest jako argument wywołania funkcji. Lista powinna zawierać: nazwę klienta i adres. Przetestuj działanie funkcji.

```
CREATE OR REPLACE FUNCTION lista(in arg1 VARCHAR(50))
RETURNS TABLE (r_nazwa VARCHAR(50), r_ulica VARCHAR(50)) AS
$$
BEGIN
RETURN QUERY
        SELECT
                nazwa, ulica
        FROM klienci
                WHERE miejscowosc = arg1;
END;
$$
LANGUAGE plpgsql;

SELECT lista('Warszawa');
```

⋆Zadanie 11.7

Napisz funkcję **rabat** obliczającą rabat jaki otrzymuje klient **kwiaciarni** składający zamówienie. Funkcję utwórz w schemacie **kwiaciarnia**. Rabat wyliczany jest na podstawie zamówień bieżących (tabela **zamowienia**) i z ostatnich siedmiu dni (tabela **historia**) w sposób następujący:

1. 5 % jeśli wartość zamówień jest większa od 0 lecz nie większa od 100 zł;
2. 10 % jeśli wartość zamówień jest z przedziału 101-400 zł;
3. 15 % jeśli wartość zamówień jest z przedziału 401-700 zł;
4. 20 % jeśli wartość zamówień jest większa od 700 zł.

Przetestuj działanie funkcji.

```
CREATE OR REPLACE FUNCTION kwiaciarnia.rabat(in arg1 VARCHAR(10))
RETURNS INTEGER AS
$$
DECLARE wynik INTEGER;
DECLARE wartoscZamowien NUMERIC(7, 2);
DECLARE wartoscHistorii NUMERIC(7, 2);
DECLARE wartoscSuma NUMERIC(7, 2);
BEGIN
        SELECT SUM(cena) INTO wartoscZamowien
        FROM
                kwiaciarnia.zamowienia
        WHERE idklienta = arg1;

        SELECT SUM(cena) INTO wartoscHistorii
        FROM
                kwiaciarnia.historia
        WHERE idklienta = arg1 AND termin > CURRENT_DATE - interval '7 days';

        wartoscSuma := wartoscZamowien + wartoscHistorii;

        IF wartoscZamowien BETWEEN 1 AND 100 THEN
                wynik := 5;
        ELSIF wartoscZamowien BETWEEN 101 AND 400 THEN
                wynik := 10;
        ELSIF wartoscZamowien BETWEEN 401 AND 700 THEN
                wynik := 15;
        ELSIF wartoscZamowien > 700 THEN
                wynik := 20;
        ELSE
                wynik := 0;
        END IF;

        RETURN wynik;
END;
$$ LANGUAGE PLpgSQL;

SELECT kwiaciarnia.rabat('msowins');
```