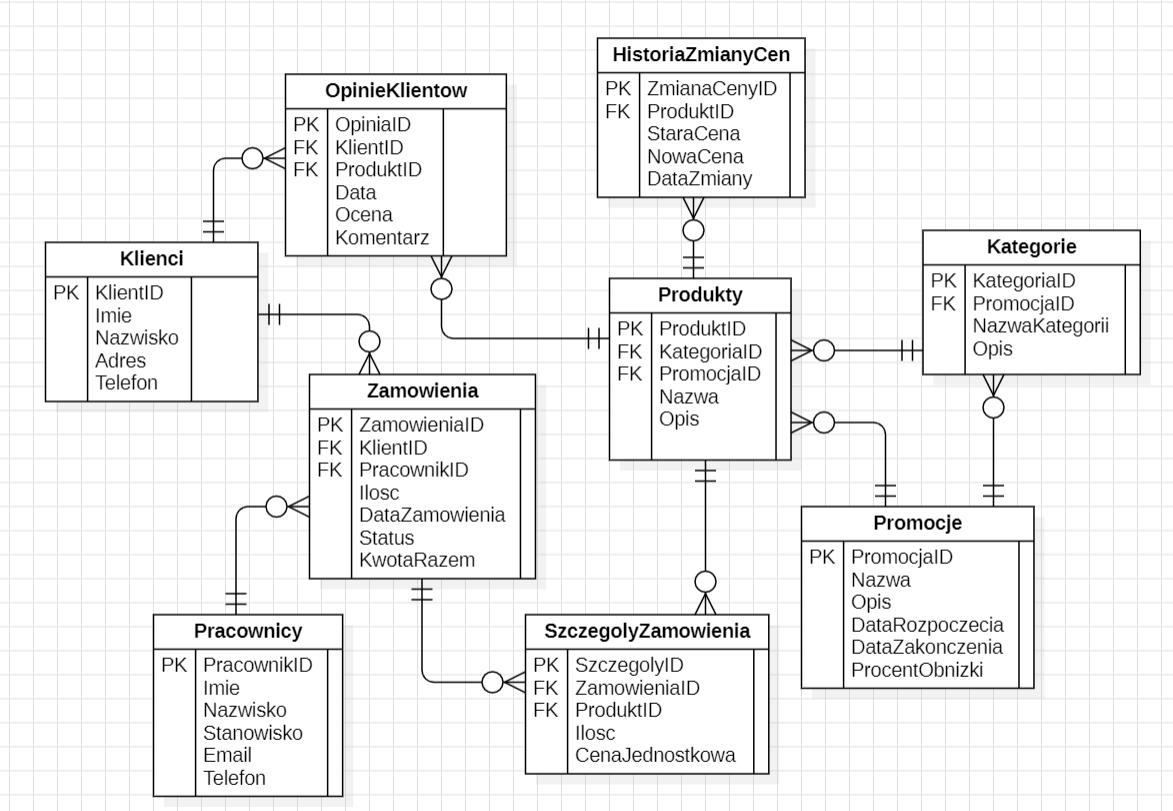


**Opis Projektu:**

Baza danych dla sklepu internetowego, która umożliwia śledzenie i analizowanie kluczowych aspektów działalności firmy. Pozwala na:

* Zarządzanie informacjami o klientach, w tym ich danymi kontaktowymi oraz historią zamówień.
* Katalogowanie produktów, z podziałem na kategorie, oraz śledzenie informacji o nich, takich jak ceny i opisy.
* Rejestrowanie i analizowanie promocji, które mogą mieć wpływ na sprzedaż.
* Zbieranie i analizowanie opinii klientów na temat produktów, co może dostarczyć cennych informacji zwrotnych.
* Śledzenie informacji o pracownikach, w tym ich danych kontaktowych i stanowisk, co może być przydatne w zarządzaniu zasobami ludzkimi.
* Monitorowanie historii zmian cen produktów, co umożliwia analizę strategii cenowej firmy.

**Diagram ERD:**



**Opis bazy danych:**

Tabela Klienci:  
Ta tabela przechowuje informacje o klientach, takie jak imię, nazwisko, adres e-mail i numer telefonu. Każdy klient ma unikalny identyfikator KlientID, który służy jako klucz główny.

Tabela Produkty:  
Tabela ta zawiera listę produktów oferowanych przez firmę. Każdy produkt ma unikalny identyfikator ProduktID, nazwę, opis oraz powiązanie z kategorią produktu (KategoriaID).

Tabela Kategorie:  
Ta tabela przechowuje informacje o kategoriach produktów. Każda kategoria ma unikalny identyfikator KategoriaID, nazwę oraz opis.

Tabela Zamowienia:  
Tabela ta rejestruje wszystkie złożone zamówienia. Każde zamówienie ma unikalny identyfikator ZamowienieID, powiązanie z klientem (KlientID), datę zamówienia, status oraz kwotę razem za całe zamówienie.

Tabela SzczegolyZamowienia:  
Ta tabela przechowuje szczegóły każdego zamówienia, takie jak produkty, ilości i ceny jednostkowe. Każdy wiersz ma unikalny identyfikator SzczegolyID, powiązanie z zamówieniem (ZamowienieID) oraz produktem (ProduktID), ilość zamówionych produktów oraz cenę jednostkową.

Tabela Promocje:  
Tabela ta zawiera informacje o aktualnych promocjach. Każda promocja ma unikalny identyfikator PromocjaID, nazwę, opis, daty rozpoczęcia i zakończenia oraz procent obniżki.

Tabela OpinieKlientow:  
Ta tabela przechowuje opinie klientów na temat produktów. Każda opinia ma unikalny identyfikator OpiniaID, powiązanie z klientem (KlientID) i produktem (ProduktID), datę, ocenę oraz komentarz.

Tabela Pracownicy:  
Tabela ta zawiera informacje o pracownikach firmy, takich jak imię, nazwisko, stanowisko, adres e-mail i numer telefonu. Każdy pracownik ma unikalny identyfikator PracownikID.

Tabela HistoriaZmianCen:  
Ta tabela rejestruje historię zmian cen produktów. Każda zmiana ceny ma unikalny identyfikator ZmianaCenyID, powiązanie z produktem (ProduktID), starą cenę, nową cenę oraz datę zmiany.

**Powiązania:**

Produkty mają powiązanie jeden do wielu z HistoriaZmianCen, gdzie jeden produkt może mieć wiele wpisów w historii zmian cen.

Kategorie mają powiązanie jeden do wielu z Produktami, gdzie jedna kategoria może zawierać wiele produktów.

Promocje mają powiązanie jeden do wielu z Kategoriami i Produktami, gdzie jedna promocja może obejmować wiele kategorii lub produktów.

Produkty mają powiązanie jeden do wielu z SzczegolyZamowienia, gdzie jeden produkt może występować w wielu szczegółach zamówień.

Zamowienia mają powiązanie jeden do wielu z SzczegolyZamowienia, gdzie jedno zamówienie może zawierać wiele szczegółów.

Pracownicy mają powiązanie jeden do wielu z Zamowieniami, gdzie jeden pracownik może obsługiwać wiele zamówień.

Klienci mają powiązanie jeden do wielu z Zamowieniami i OpinieKlientow, gdzie jeden klient może złożyć wiele zamówień i wystawić wiele opinii.

**Funkcjonalność:**

Procedury składowane do zarządzania zamówieniami:

* Procedura do składania nowego zamówienia, która aktualizuje tabele Zmówienia i szczegolyZamowienia.
* Procedura do anulowania lub modyfikacji istniejącego zamówienia.
* Procedura do aktualizacji statusu zamówienia.

Funkcje do obliczania statystyk sprzedaży:

* Funkcja obliczająca średnią ocenę produktu na podstawie opinii klientów.
* Automatyczne obliczanie KwotyRazem dla zamówień na podstawie szczegółów zamówienia.
* Generowanie raportów sprzedaży

Funkcje do obsługi promocji:

* Funkcja obliczająca cenę produktu po zastosowaniu obowiązującej promocji.
* Procedura do tworzenia nowej promocji i aktualizacji cen produktów objętych promocją.
* Wyzwalacz automatycznie ustawiający cenę produktu na podstawie podanej promocji.

**Użyte technologie:**

Do wykonania projektu użyliśmy następujących technologii:

* **Oracle 19c** - zaawansowana relacyjna baza danych, która zapewnia niezawodność i skalowalność, konieczne do zarządzania dużymi ilościami danych.
* **Java** - współbieżny, oparty na klasach, obiektowy język programowania ogólnego zastosowania. Zaprojektowaliśmy w nim GUI do naszego projektu.

**Procedury:**

Dla tabel: klient, pracownik, promocje, opinie klientów, produkt i kategorie zaimplementowaliśmy operacje CRUD

Przykładowe operacje CRUD:

CREATE OR REPLACE PROCEDURE dodaj\_nowego\_pracownika(

p\_imie IN VARCHAR2,

p\_nazwisko IN VARCHAR2,

p\_stanowisko IN VARCHAR2,

p\_email IN VARCHAR2,

p\_telefon IN VARCHAR2

)

IS

nowy\_pracownik\_id NUMBER;

BEGIN

SELECT MAX(PracownikID) + 1 INTO nowy\_pracownik\_id FROM Pracownicy;

INSERT INTO Pracownicy (PracownikID, Imie, Nazwisko, Stanowisko, Email, Telefon)

VALUES (nowy\_pracownik\_id, p\_imie, p\_nazwisko, p\_stanowisko, p\_email, p\_telefon);

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Nowy pracownik dodany. PracownikID: ' || nowy\_pracownik\_id);

EXCEPTION

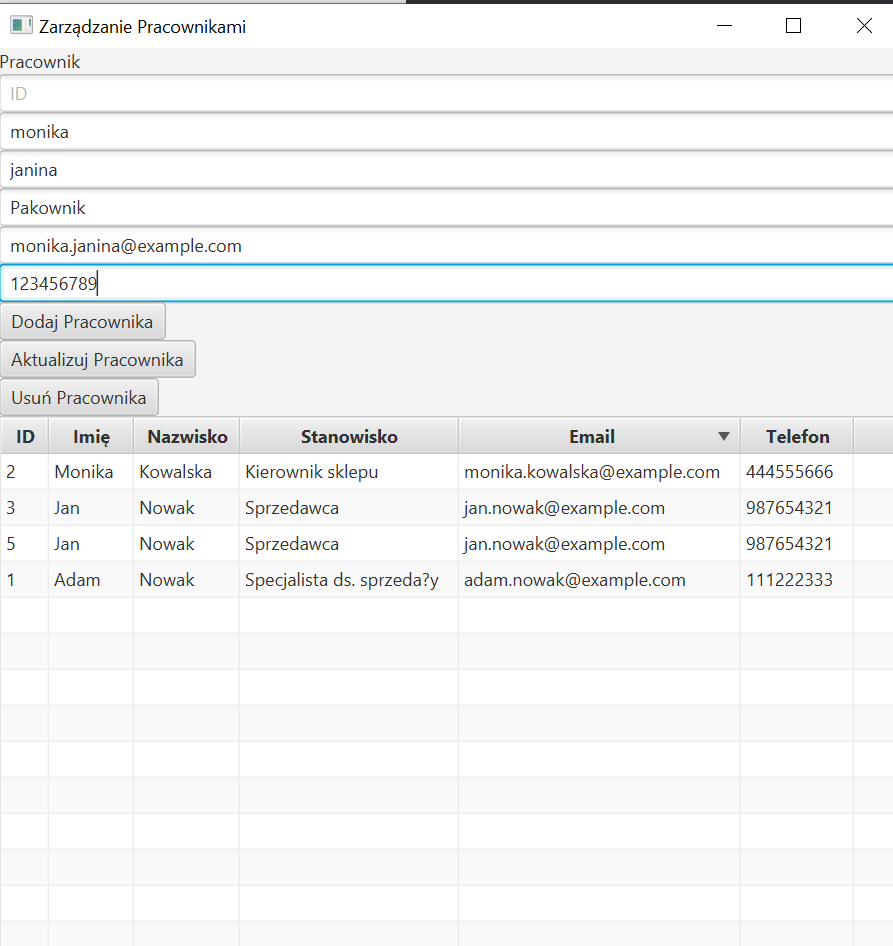
WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('B��d: ' || SQLERRM);

ROLLBACK;

END dodaj\_nowego\_pracownika;

/



-- Procedura do usuwania pracownika wraz z powi�zanymi danymi

CREATE OR REPLACE PROCEDURE usun\_pracownika\_z\_zaleznosci (

p\_pracownik\_id IN NUMBER

)

IS

BEGIN

-- Usuwanie zam�wie� powi�zanych z pracownikiem

DELETE FROM SzczegolyZamowienia

WHERE ZamowieniaID IN (

SELECT ZamowienieID

FROM Zamowienia

WHERE PracownikID = p\_pracownik\_id

);

DELETE FROM Zamowienia

WHERE PracownikID = p\_pracownik\_id;

-- Usuwanie pracownika

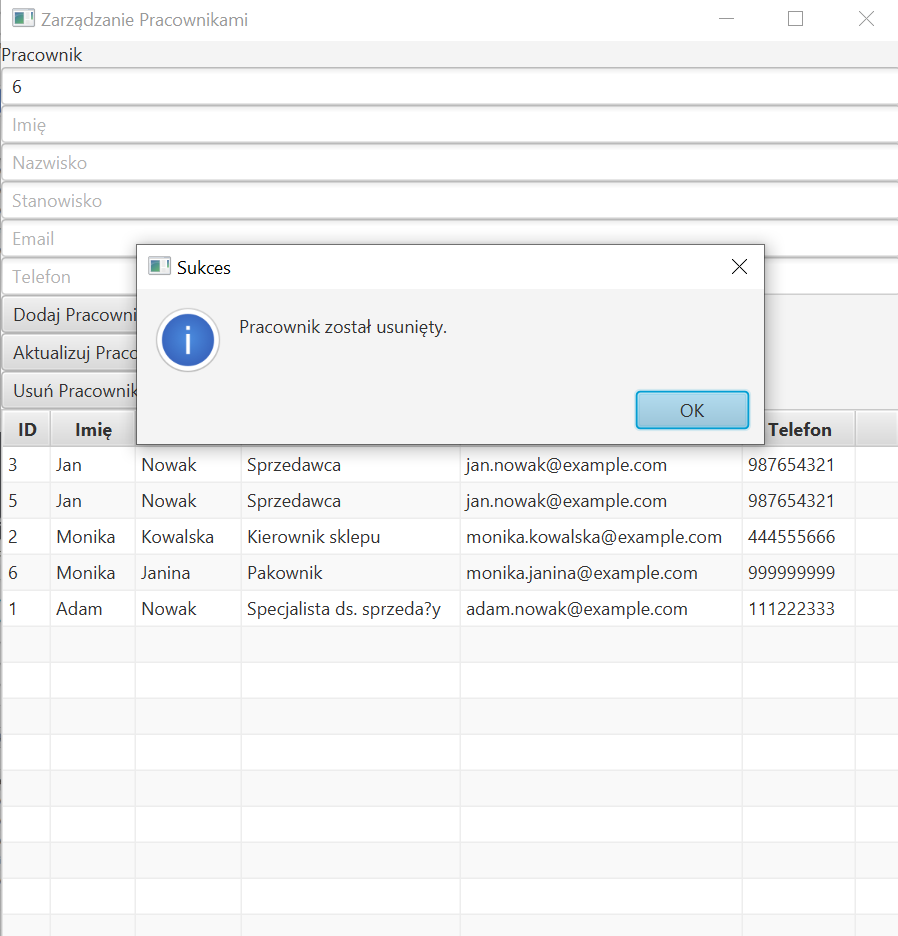
DELETE FROM Pracownicy

WHERE PracownikID = p\_pracownik\_id;

COMMIT;

END usun\_pracownika\_z\_zaleznosci;

/



CREATE OR REPLACE PROCEDURE wyswietl\_pracownikow

IS

CURSOR kursor\_pracownikow IS

SELECT PracownikID, Imie, Nazwisko, Stanowisko, Email, Telefon

FROM Pracownicy;

v\_pracownik Pracownicy%ROWTYPE;

BEGIN

OPEN kursor\_pracownikow;

LOOP

FETCH kursor\_pracownikow INTO v\_pracownik;

EXIT WHEN kursor\_pracownikow%NOTFOUND;

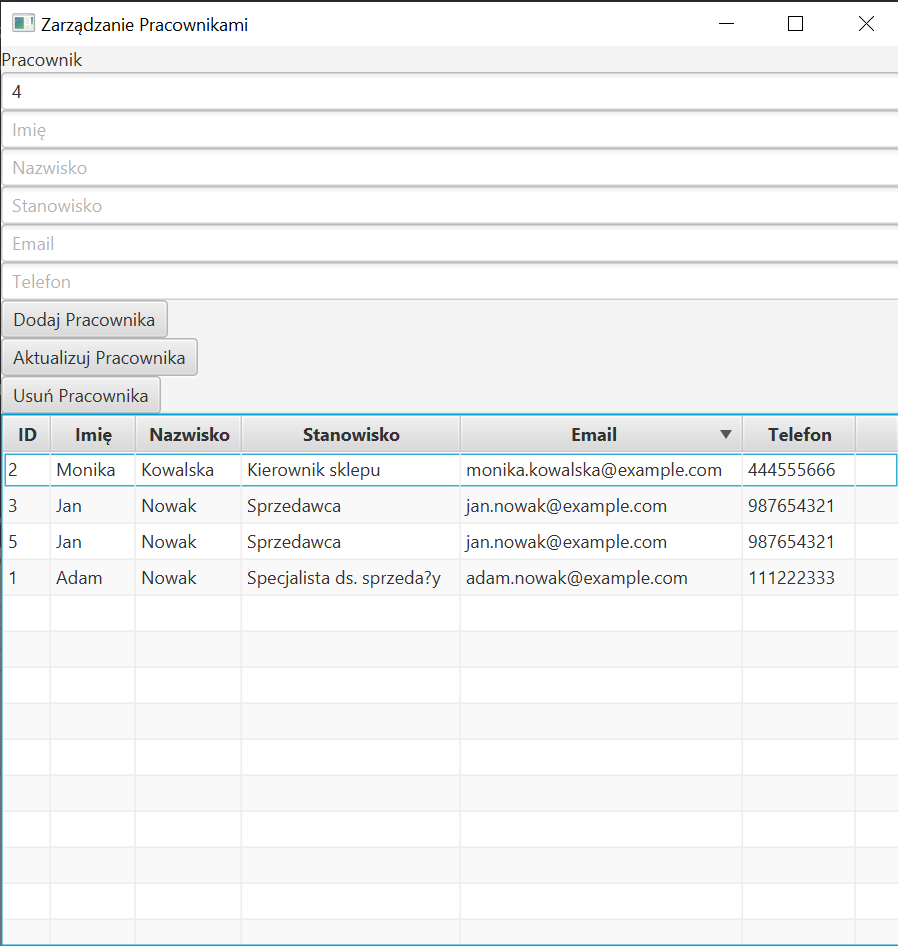
DBMS\_OUTPUT.PUT\_LINE(v\_pracownik.PracownikID || ' ' || v\_pracownik.Imie || ' ' || v\_pracownik.Nazwisko || ' ' || v\_pracownik.Stanowisko || ' ' || v\_pracownik.Email || ' ' || v\_pracownik.Telefon);

END LOOP;

CLOSE kursor\_pracownikow;

END wyswietl\_pracownikow;

/



CREATE OR REPLACE PROCEDURE aktualizuj\_pracownika(

p\_pracownik\_id IN NUMBER,

p\_imie IN VARCHAR2,

p\_nazwisko IN VARCHAR2,

p\_stanowisko IN VARCHAR2,

p\_email IN VARCHAR2,

p\_telefon IN VARCHAR2

)

IS

BEGIN

UPDATE Pracownicy

SET Imie = p\_imie,

Nazwisko = p\_nazwisko,

Stanowisko = p\_stanowisko,

Email = p\_email,

Telefon = p\_telefon

WHERE PracownikID = p\_pracownik\_id;

COMMIT;

EXCEPTION

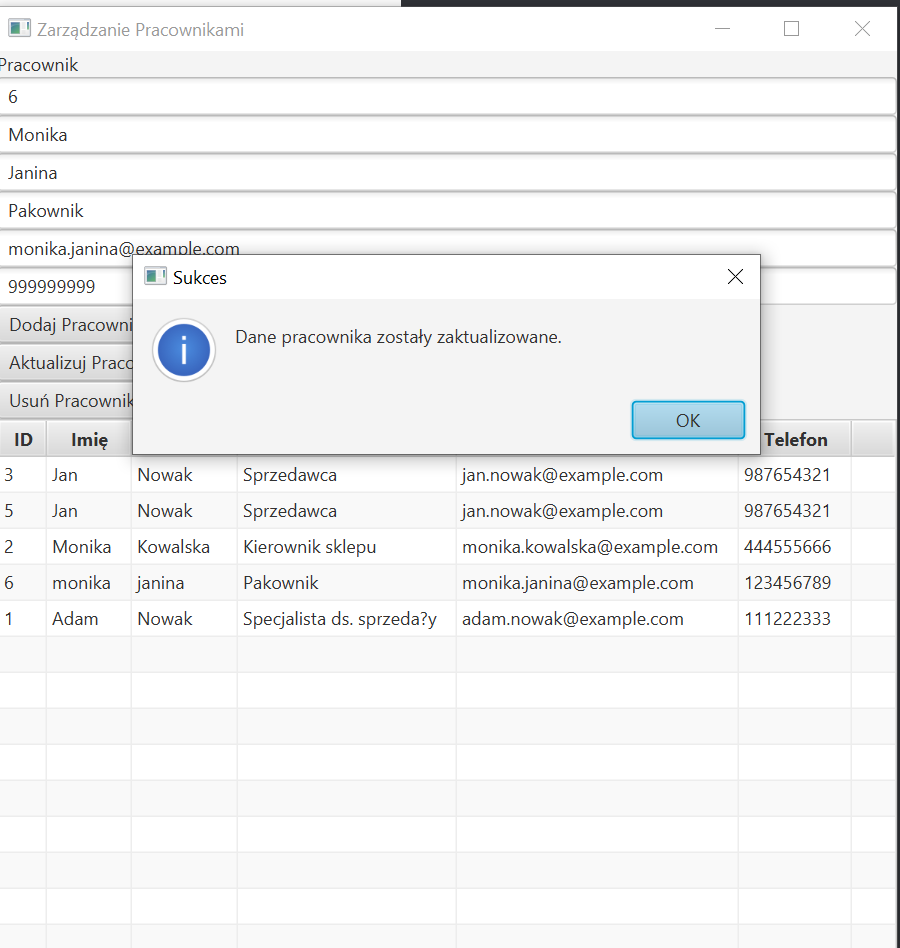
WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('B��d: ' || SQLERRM);

ROLLBACK;

END aktualizuj\_pracownika;

/



**Funkcje:**

* **Składanie zamówień:**

- funkcja odpowiedzialna za składanie zamówień do bazy danych. Pobiera id klienta, pracownika i produktu oraz ilość produktu, następnie za pomocą id produktu, pobiera jego cenę. Potem insertuje te informacje do tabeli zamówień, gdzie KwotaRazem to ilość produktu razy jego cena. Na samym końcu dodaje też wpis do tabeli SzczegolyZamowien

-- Skrypt do tworzenia procedur i funkcji

CREATE OR REPLACE PROCEDURE dodaj\_zamowienie(

p\_klient\_id IN NUMBER,

p\_pracownik\_id IN NUMBER,

p\_produkt\_id IN NUMBER,

p\_ilosc IN NUMBER

)

IS

nowy\_zamowienie\_id NUMBER;

cena\_jednostkowa NUMBER;

BEGIN

SELECT CenaJednostkowa INTO cena\_jednostkowa FROM Produkty WHERE ProduktID = p\_produkt\_id;

SELECT SEQ\_Zamowienia.NEXTVAL INTO nowy\_zamowienie\_id FROM dual;

INSERT INTO Zamowienia (ZamowienieID, KlientID, PracownikID, Ilosc, DataZamowienia, Status, KwotaRazem)

VALUES (nowy\_zamowienie\_id, p\_klient\_id, p\_pracownik\_id, p\_ilosc, SYSDATE, NULL, p\_ilosc \* cena\_jednostkowa);

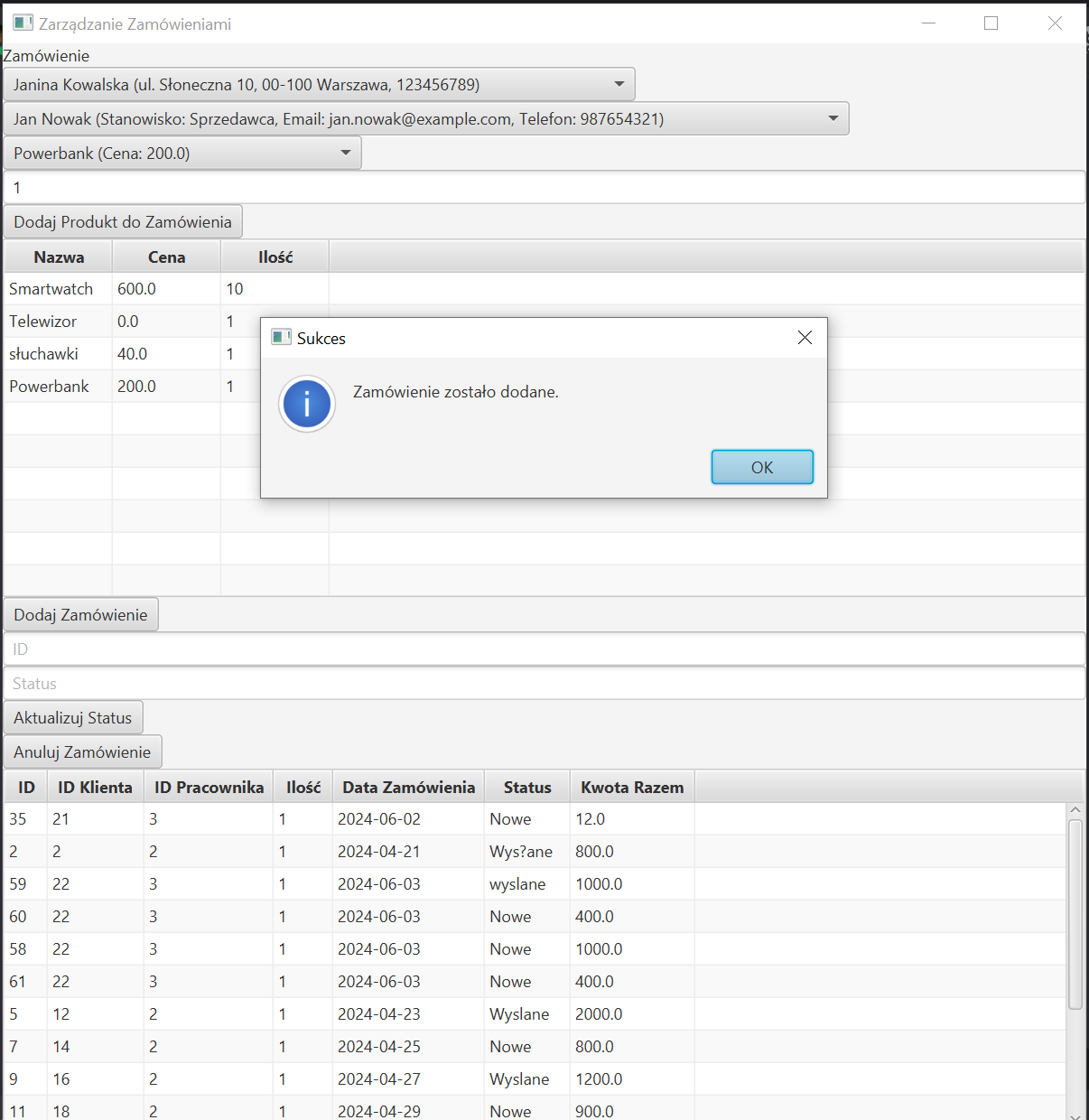
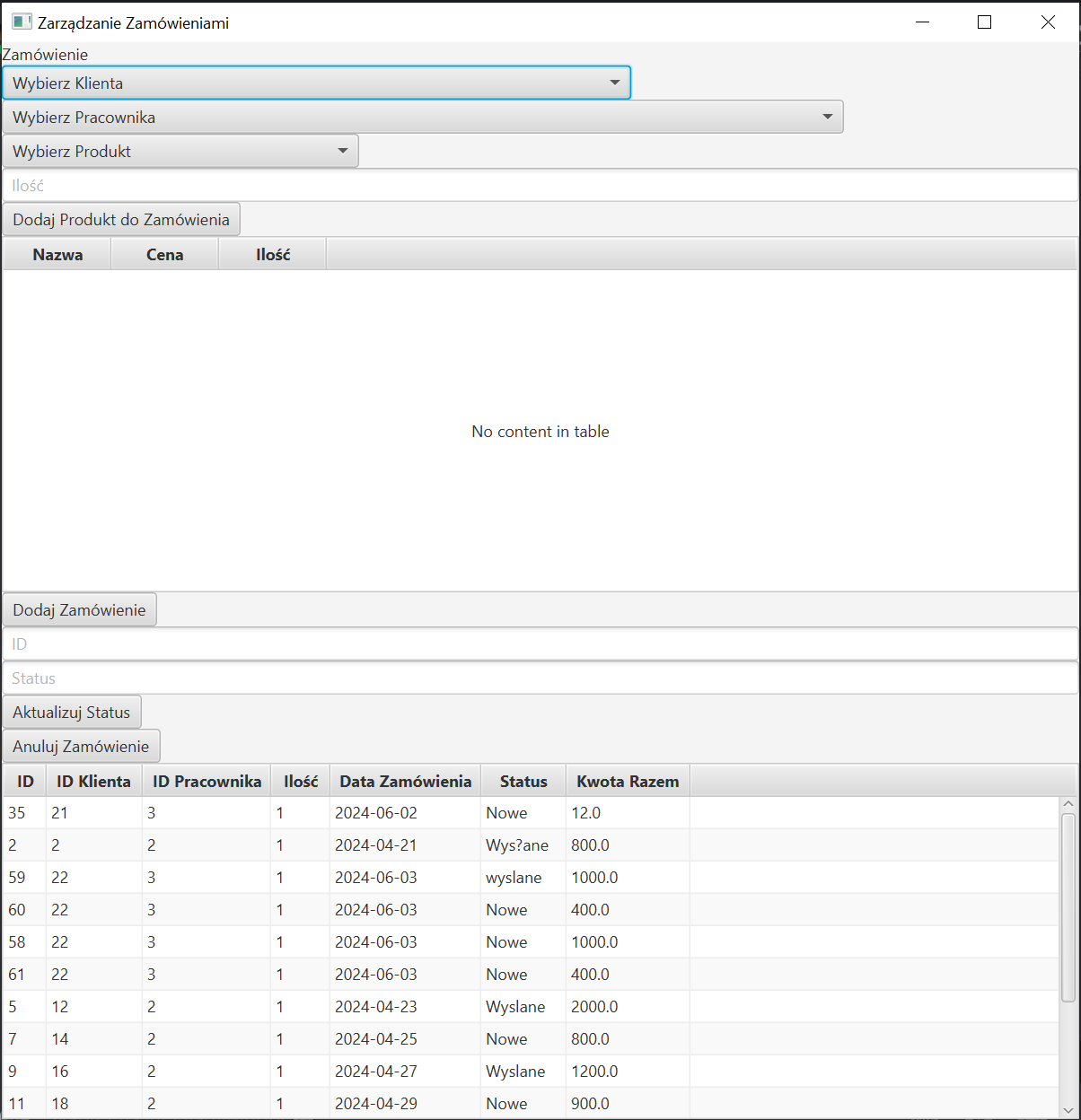
INSERT INTO SzczegolyZamowienia (SzczegolyID, ZamowieniaID, ProduktID, Ilosc, CenaJednostkowa)

VALUES (SEQ\_SzczegolyZamowienia.NEXTVAL, nowy\_zamowienie\_id, p\_produkt\_id, p\_ilosc, cena\_jednostkowa);

COMMIT;

END dodaj\_zamowienie;

/



* **Anulowanie zamówień:**

-funkcja odpowiedzialna za anulowanie zamówień. Pobiera id zamówienia i na jego podstawie usuwa zamówienie z tabeli Zamowienia i SzczegolyZamowien

CREATE OR REPLACE PROCEDURE anuluj\_zamowienie(

p\_zamowienie\_id IN NUMBER

)

IS

BEGIN

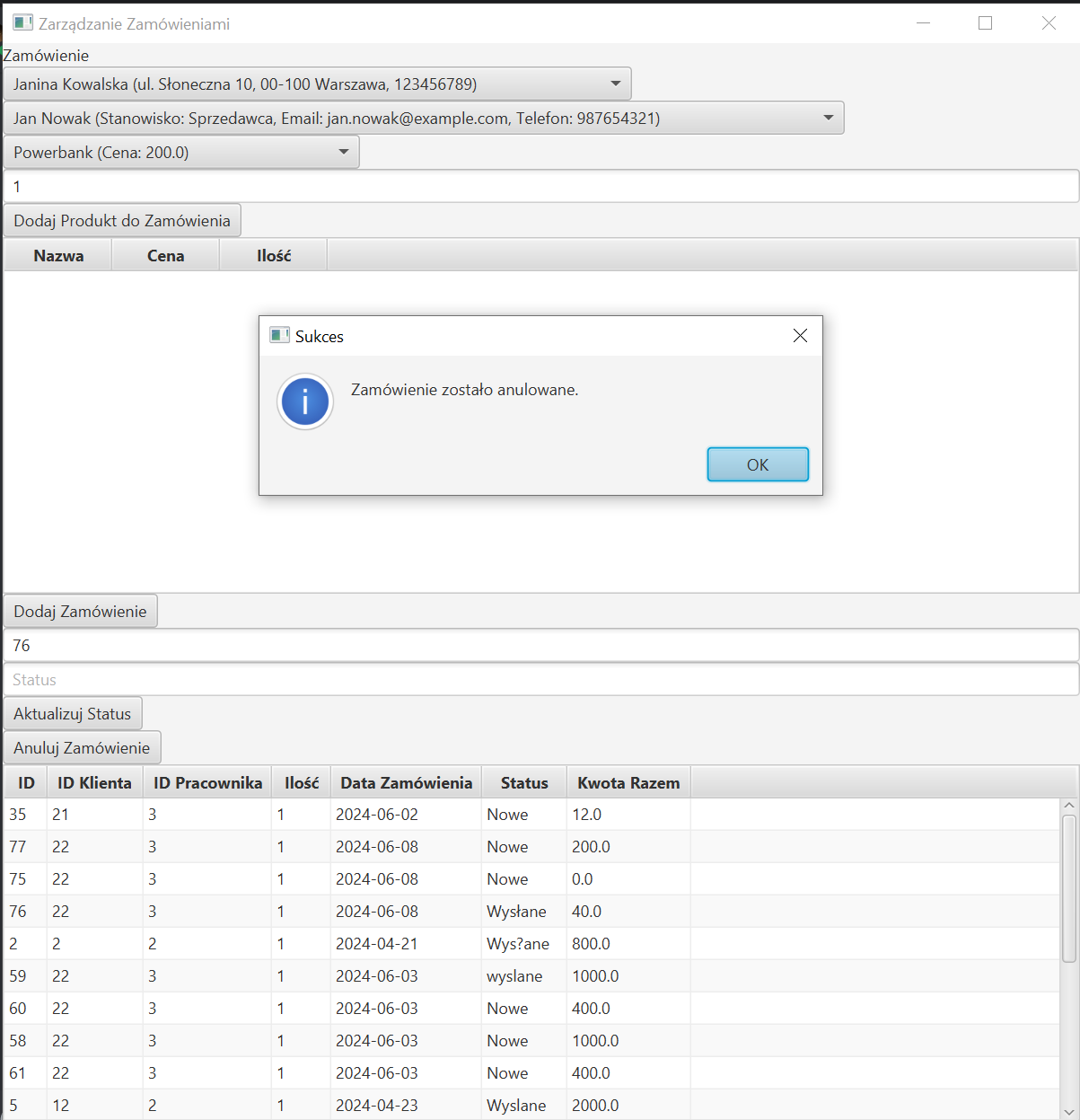
DELETE FROM SzczegolyZamowienia WHERE ZamowieniaID = p\_zamowienie\_id;

DELETE FROM Zamowienia WHERE ZamowienieID = p\_zamowienie\_id;

COMMIT;

END anuluj\_zamowienie;

/



* **Aktualizowanie statusu zamówienia:**

-funkcja, która aktualizuje status zamówienia. Pobiera id zamówienia i nowy status dla zamówienia. Następnie aktualizuje kolumnę Status dla zamówienia o wcześniej pobranym id.

CREATE OR REPLACE PROCEDURE aktualizuj\_status\_zamowienia(

p\_zamowienie\_id IN NUMBER,

p\_nowy\_status IN VARCHAR2

)

IS

BEGIN

UPDATE Zamowienia

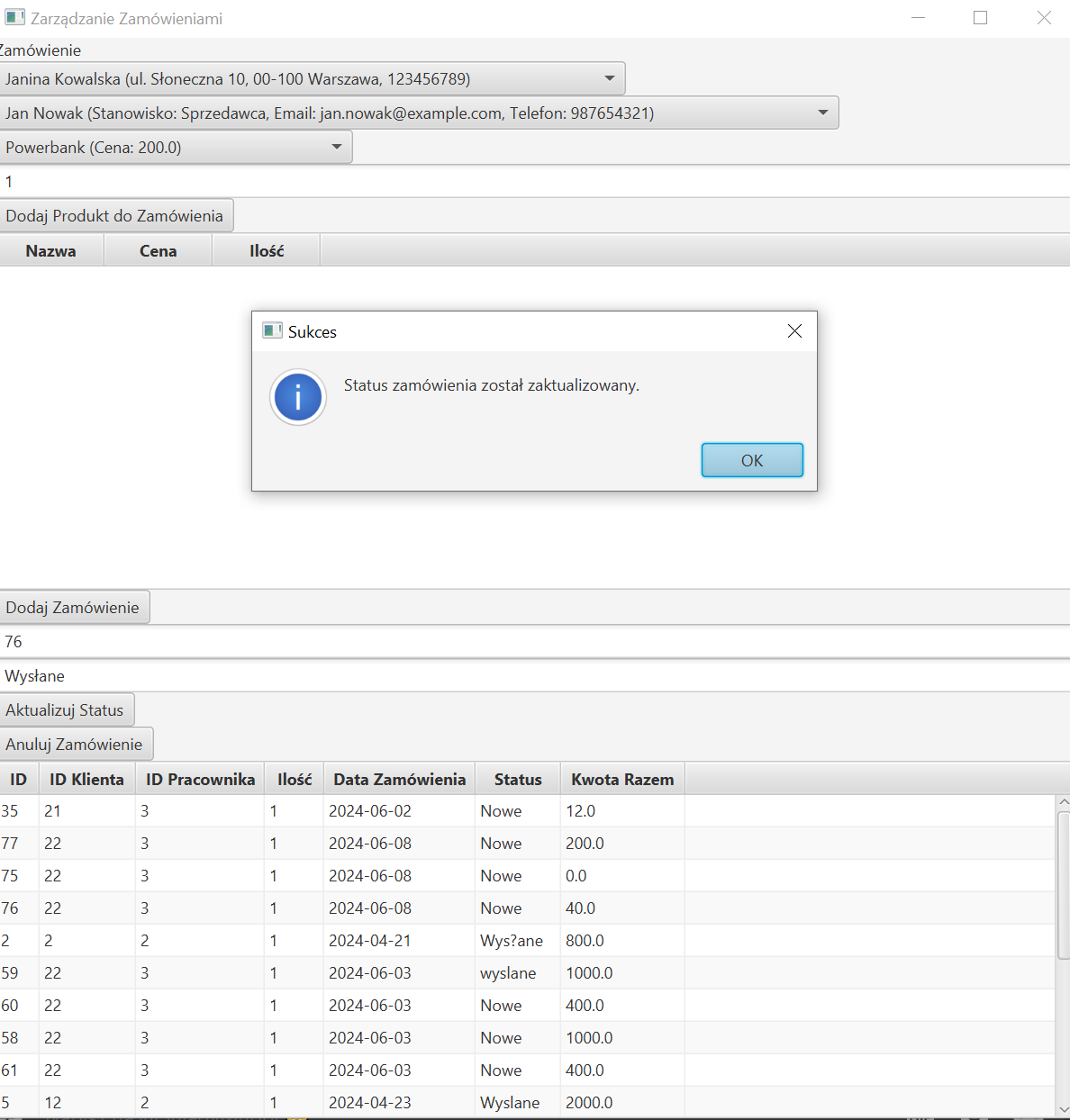
SET Status = p\_nowy\_status

WHERE ZamowienieID = p\_zamowienie\_id;

COMMIT;

END aktualizuj\_status\_zamowienia;

/



* **Wyświetlanie średniej oceny produktu:**

-funkcja, która oblicza średnią ocenę produktu, na podstawie jej opinii. Jest używana w raportach sprzedaży.

CREATE OR REPLACE FUNCTION srednia\_ocena\_produktu(

p\_produkt\_id IN NUMBER

) RETURN NUMBER

IS

v\_srednia\_ocena NUMBER;

BEGIN

SELECT AVG(Ocena)

INTO v\_srednia\_ocena

FROM OpinieKlientow

WHERE ProduktID = p\_produkt\_id;

RETURN NVL(v\_srednia\_ocena, 0);

END srednia\_ocena\_produktu;

/

* **Wyliczanie ceny po promocji:**

-funkcja, która wylicza cenę produktu po zastosowaniu dla niej promocji. Jest używana w procedurach dodawania i aktualizacji produktu.

CREATE OR REPLACE FUNCTION cena\_po\_promocji(

p\_produkt\_id IN NUMBER

) RETURN NUMBER

IS

v\_cena NUMBER;

v\_procent\_obnizki NUMBER;

BEGIN

SELECT p.CenaJednostkowa, pr.ProcentObnizki

INTO v\_cena, v\_procent\_obnizki

FROM Produkty p

JOIN Promocje pr ON p.PromocjaID = pr.PromocjaID

WHERE p.ProduktID = p\_produkt\_id;

IF v\_procent\_obnizki IS NOT NULL THEN

v\_cena := v\_cena \* (1 - v\_procent\_obnizki / 100);

END IF;

RETURN v\_cena;

END cena\_po\_promocji;

/

* **Generowanie raportów sprzedaży:**

-funkcja do generowania raportów sprzedaży. Kursorem pobiera nam informacje o produktach takie jak: Nazwa, Kategoria, Zarobek (ilość zarobionych pieniędzy za ten produkt), ilość sprzedanych produktów, średnia ocena produktów i wyświetla je.

CREATE OR REPLACE FUNCTION RaportSprzedazyProduktow

RETURN SYS\_REFCURSOR IS

rc SYS\_REFCURSOR;

BEGIN

OPEN rc FOR

SELECT

p.Nazwa,

k.NazwaKategorii,

COALESCE(SUM(sz.CenaJednostkowa \* sz.Ilosc), 0) AS Zarobek,

COALESCE(SUM(sz.Ilosc), 0) AS IloscSprzedanych,

srednia\_ocena\_produktu(p.ProduktID) AS SredniaOcena

FROM Produkty p

LEFT JOIN SzczegolyZamowienia sz ON p.ProduktID = sz.ProduktID

LEFT JOIN Zamowienia z ON sz.ZamowieniaID = z.ZamowienieID AND z.Status = 'Zrealizowane'

LEFT JOIN Kategorie k ON p.KategoriaID = k.KategoriaID

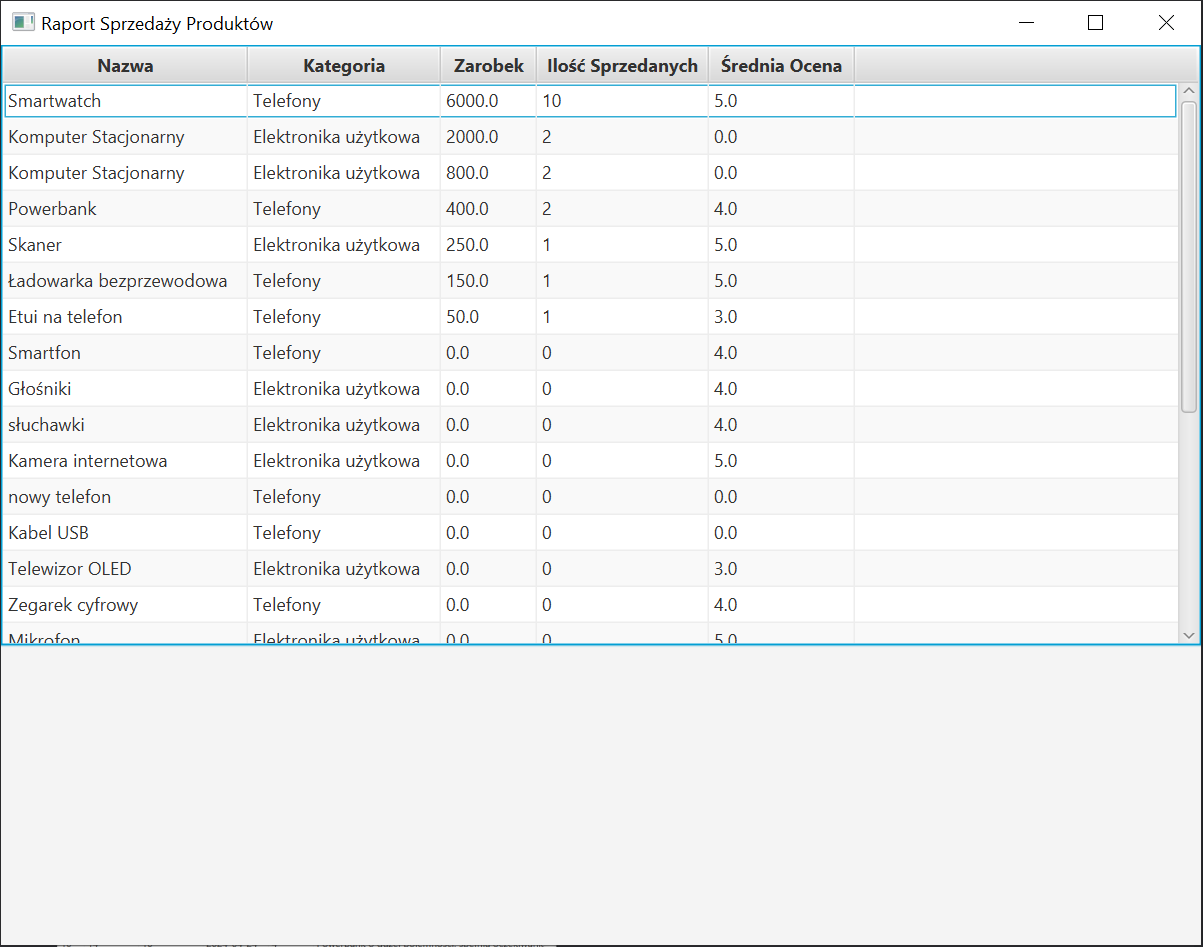
GROUP BY p.ProduktID, p.Nazwa, k.NazwaKategorii

ORDER BY Zarobek DESC;

RETURN rc;

END;

/



**Triggery:**

* **Wstawianie rekordu do HistoriZmianCen po aktualizacji ceny w produkcie:**

-trigger odpowiedzialny za aktualizacje tabeli HistoriaZmianCen za każdym razem jak CenaJednostkowa zostaje zmieniona w produkcie.

Gdy nastąpi zmiana ceny, pobiera nową i starą cenę i id produktu, a następnie wstawia rekord z tymi danymi do tabeli HistoriaZmianCen

CREATE OR REPLACE TRIGGER AktualizacjaCeny

BEFORE UPDATE OF CenaJednostkowa ON Produkty

FOR EACH ROW

DECLARE

v\_nowaCena Produkty.CenaJednostkowa%TYPE;

v\_staraCena Produkty.CenaJednostkowa%TYPE;

nowy\_zmiana\_id NUMBER;

BEGIN

v\_nowaCena := :new.CenaJednostkowa;

v\_staraCena := :old.CenaJednostkowa;

SELECT MAX(ZmianaCenyID) + 1 INTO nowy\_zmiana\_id FROM HistoriaZmianCen;

IF v\_nowaCena != v\_staraCena THEN

INSERT INTO HistoriaZmianCen (ZmianaCenyID, ProduktID, StaraCena, NowaCena, DataZmiany)

VALUES (nowy\_zmiana\_id, :old.ProduktID, v\_staraCena, v\_nowaCena, SYSDATE);

END IF;

END;

/

**Sekwencje:**

CREATE SEQUENCE SEQ\_Produkty START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE SEQ\_Kategorie START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE SEQ\_Pracownicy START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE SEQ\_Promocje START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE SEQ\_Zamowienia START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE SEQ\_SzczegolyZamowienia START WITH 1 INCREMENT BY 1; CREATE SEQUENCE SEQ\_HistoriaZmianCen START WITH 1 INCREMENT BY 1;

**Podsumowanie:**

Podsumowując, projekt stworzył kompleksowy system do składania zamówień oraz do raportowania sprzedaży, który umożliwiał użytkownikom generowanie raportów sprzedaży w sposób łatwy i efektywny, wykorzystując interfejs graficzny użytkownika w języku Java i bazę danych w języku PL/SQL.