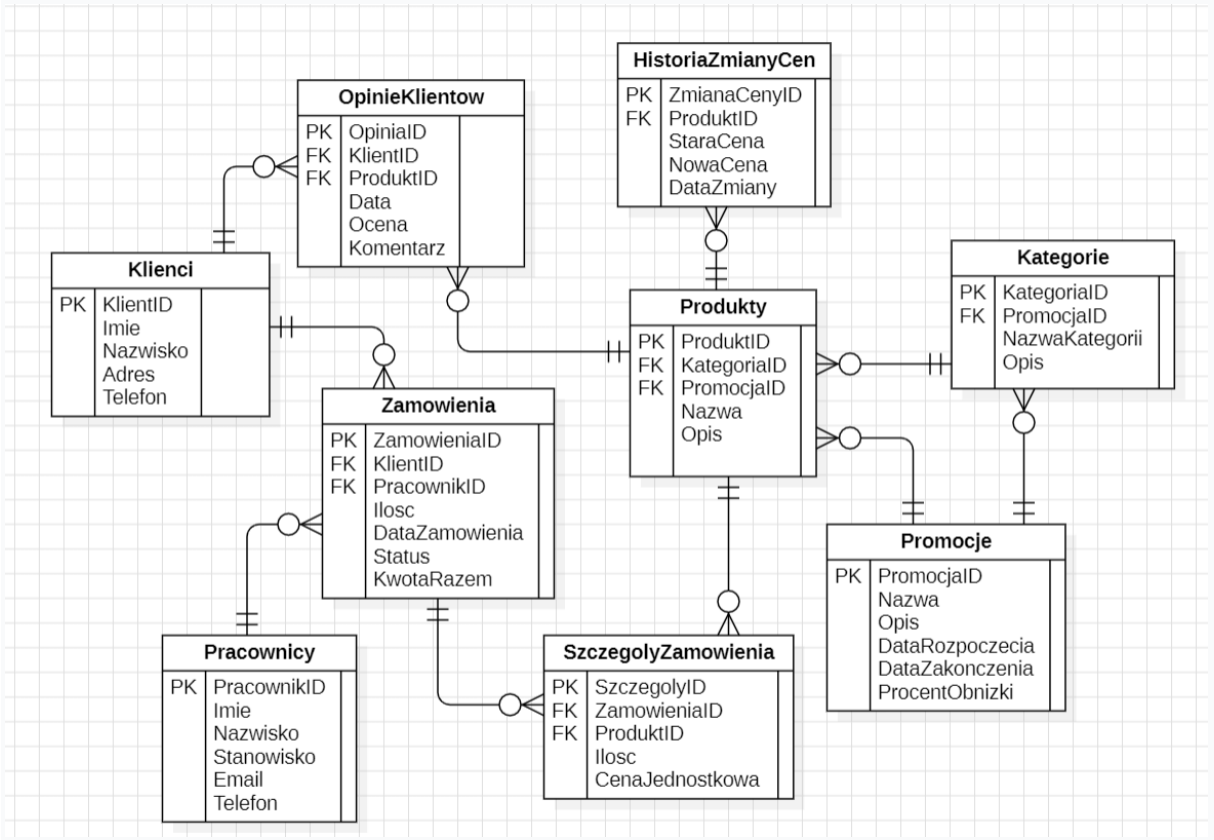
Uniwersytet Rzeszowski
Kolegium Nauk Przyrodniczych
Instytut Informatyki

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.

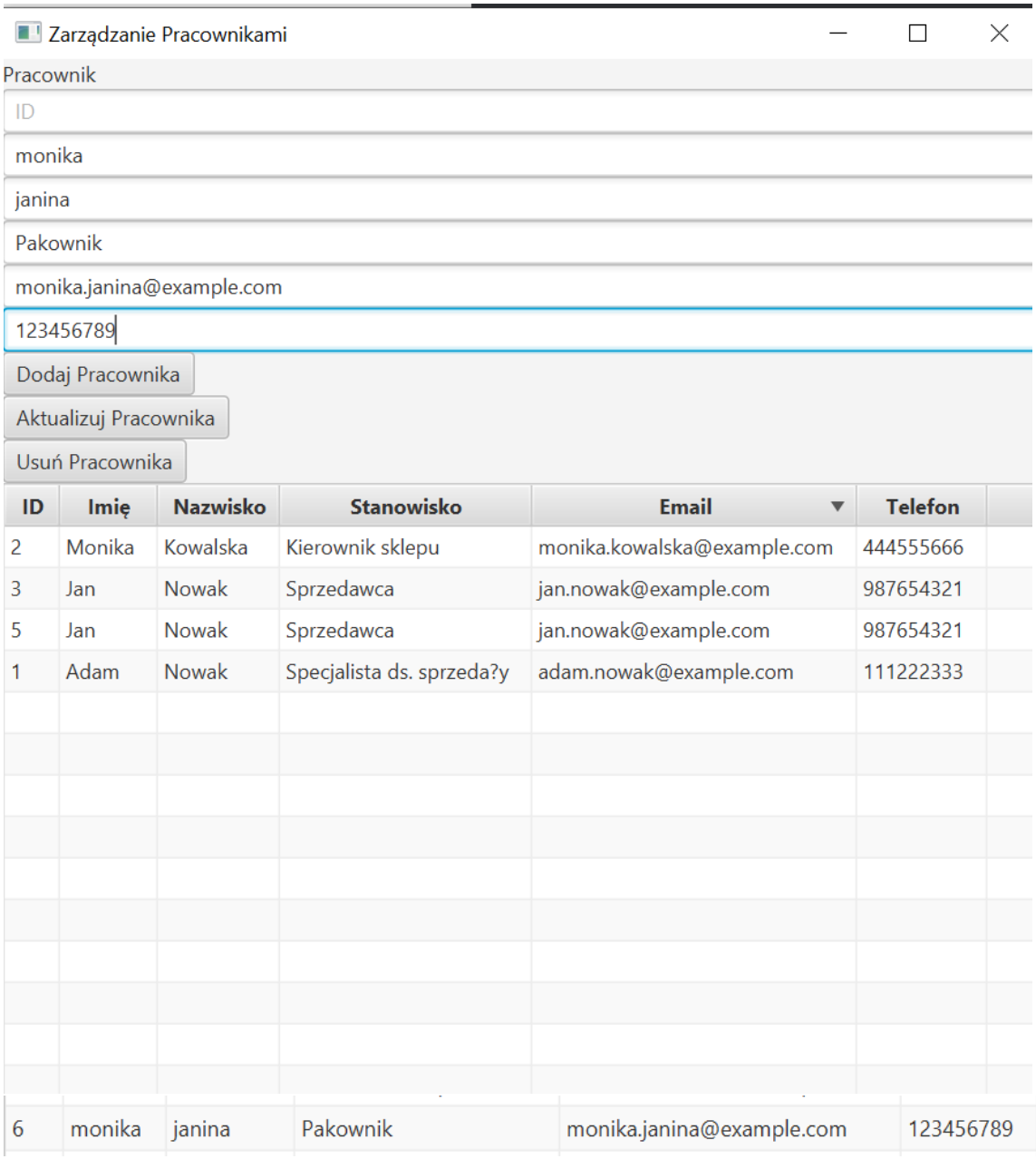
Procedury:

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

Przykładowe operacje CRUD:

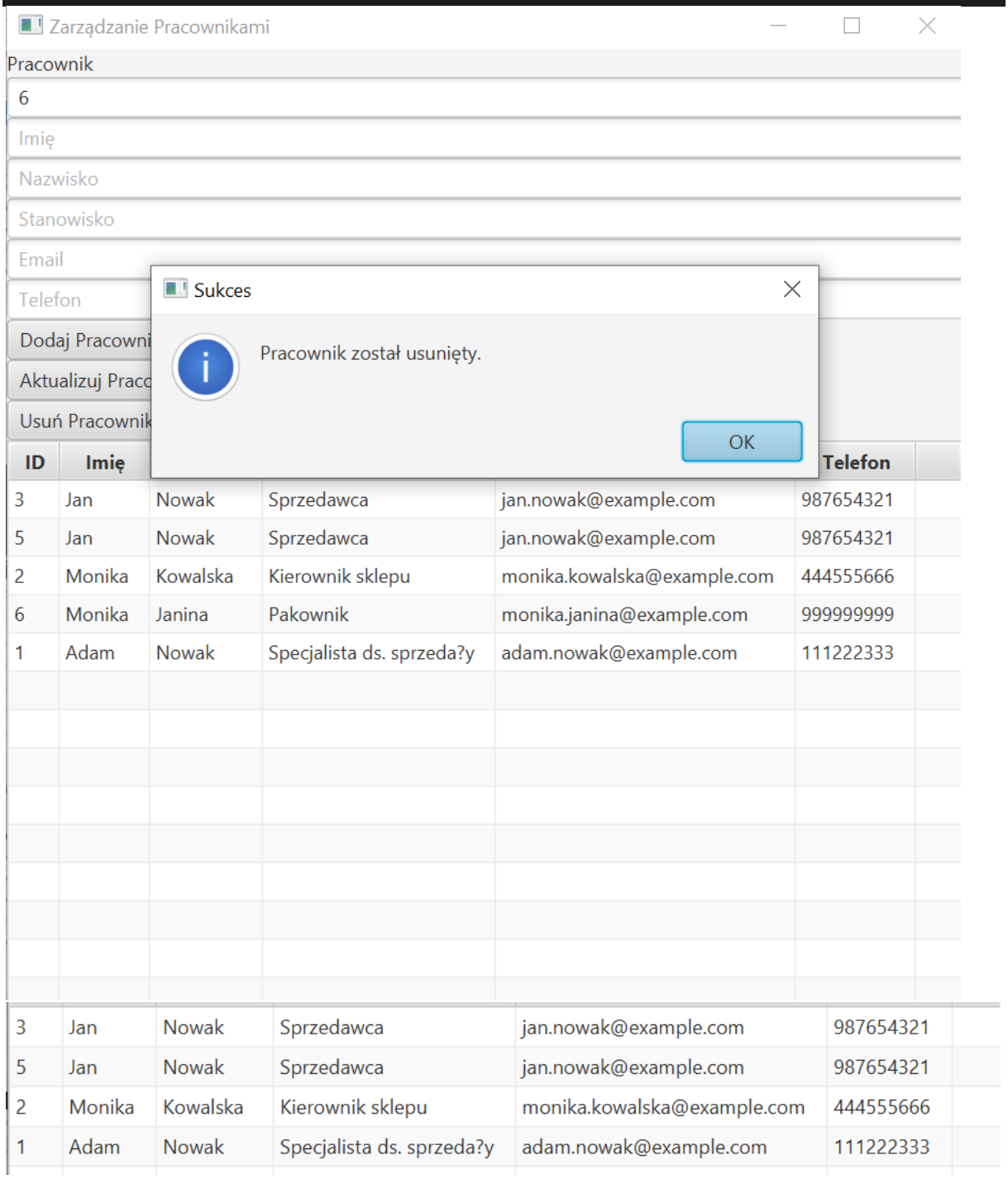
```
CREATE OR REPLACE PROCEDURE dodaj_nowego_pracownika(
```

```
    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;
/
```



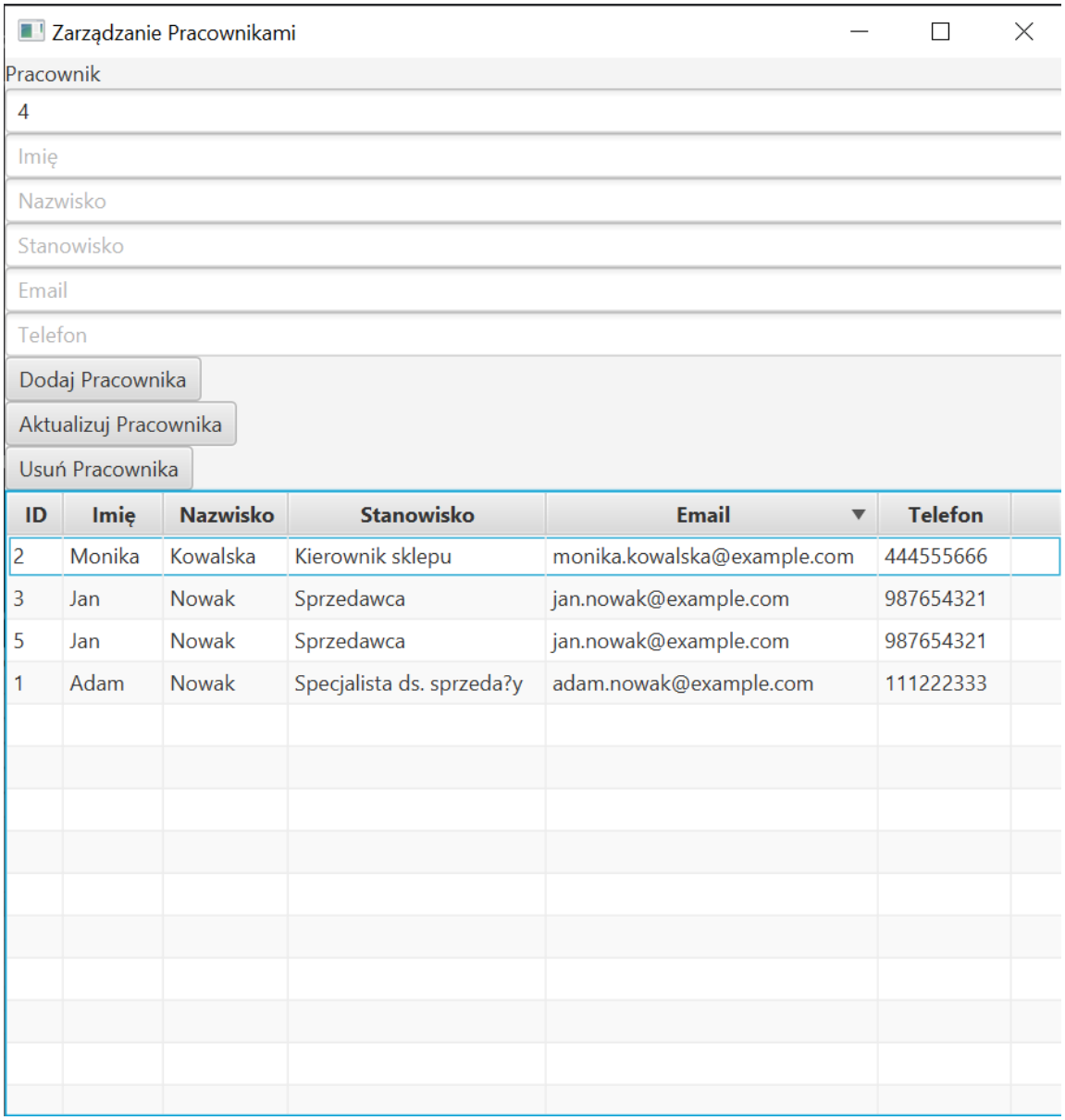

```sql
-- Procedura do usuwania pracownika wraz z powi?zanymi danymi
CREATE OR REPLACE PROCEDURE usun_pracownika_z_zaleznosci (
    p_pracownik_id IN NUMBER
)
IS
BEGIN
    -- Usuwanie zam?wie? powi?zanych z pracownikiem
    DELETE FROM SzczegolyZamowienia
    WHERE ZamowieniaID IN (
        SELECT ZamowienieID
        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;
/
```



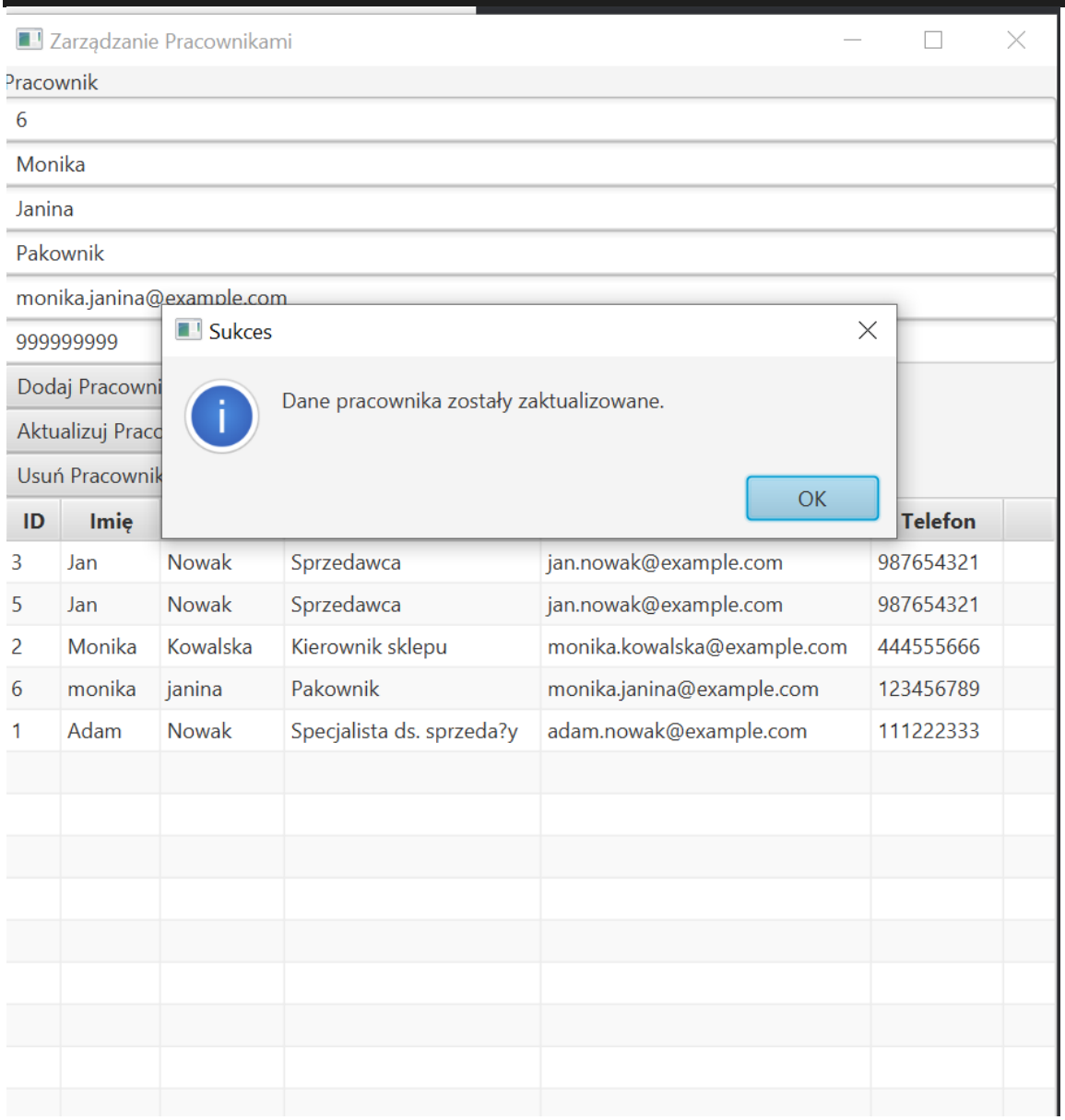
3	Jan	Nowak	Sprzedawca	jan.nowak@example.com	987654321
5	Jan	Nowak	Sprzedawca	jan.nowak@example.com	987654321
2	Monika	Kowalska	Kierownik sklepu	monika.kowalska@example.com	444555666
1	Adam	Nowak	Specjalista ds. sprzeda?y	adam.nowak@example.com	111222333

```
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;
/
```

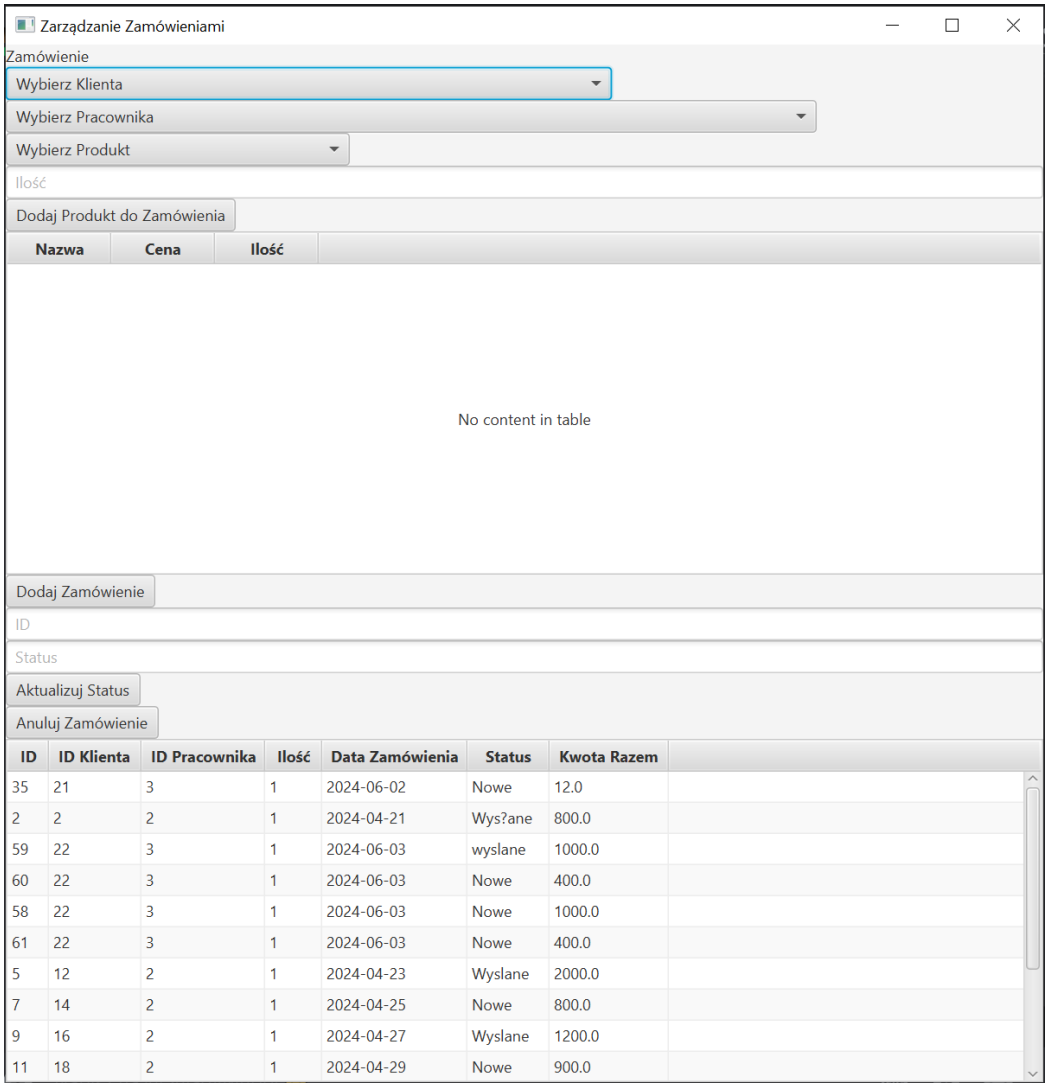


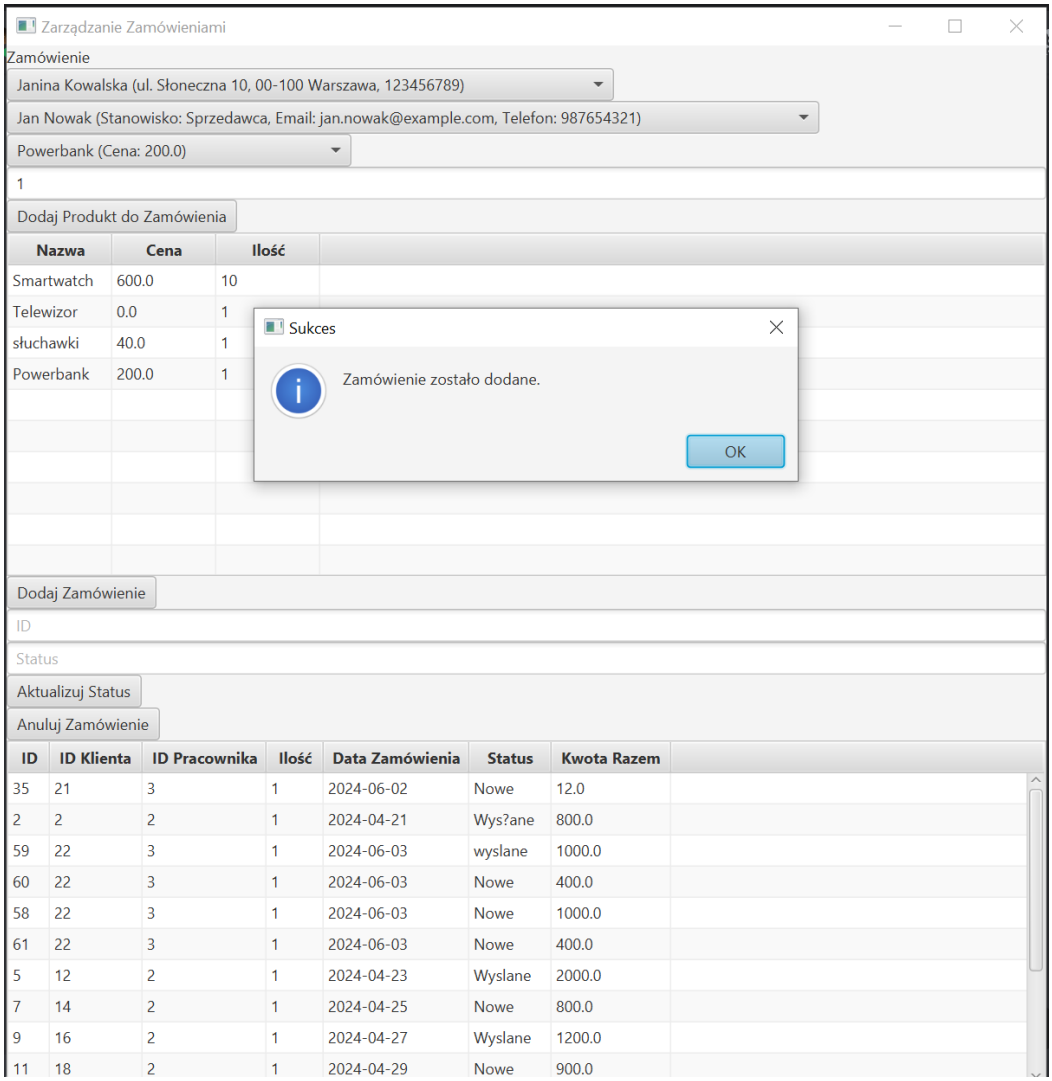
Funkcje:

- **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;
/
```

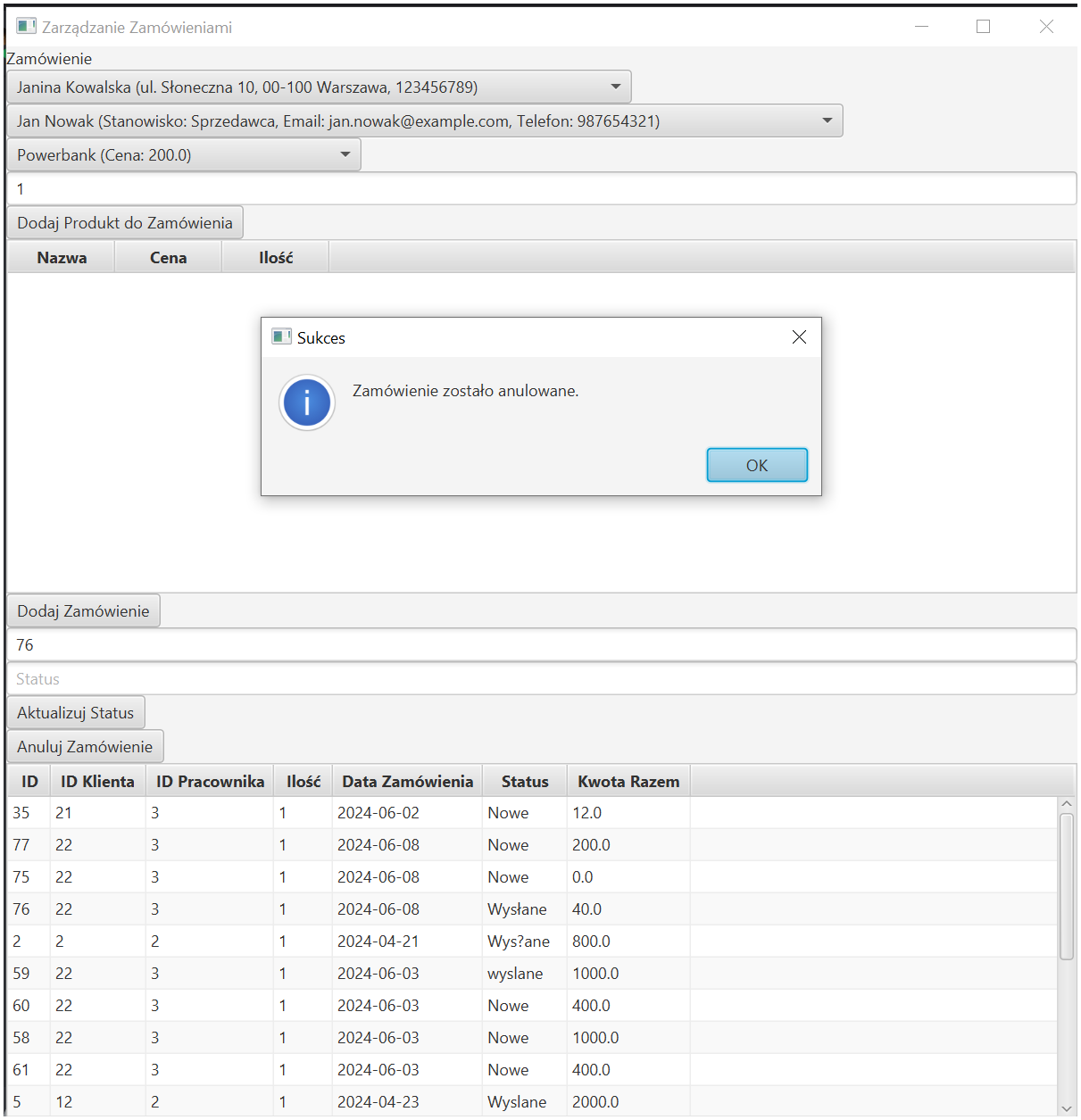

Zarządzanie Zamówieniami
Zamówienie
Wybierz Klienta
Wybierz Pracownika
Wybierz Produkt
Ilość
Dodaj Produkt do Zamówienia
Nazwa
Cena
Ilość
No content in table
Dodaj Zamówienie
ID
Status
Aktualizuj Status
Anuluj Zamówienie
ID
ID Klienta
ID Pracownika
Ilość
Data Zamówienia
Status
Kwota Razem
35 21 3 1 2024-06-02 Nowe 12.0
2 2 2 1 2024-04-21 Wys?ane 800.0
59 22 3 1 2024-06-03 wyslane 1000.0
60 22 3 1 2024-06-03 Nowe 400.0
58 22 3 1 2024-06-03 Nowe 1000.0
61 22 3 1 2024-06-03 Nowe 400.0
5 12 2 1 2024-04-23 Wyslane 2000.0
7 14 2 1 2024-04-25 Nowe 800.0
9 16 2 1 2024-04-27 Wyslane 1200.0
11 18 2 1 2024-04-29 Nowe 900.0



- **Anulowanie zamówień:**

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

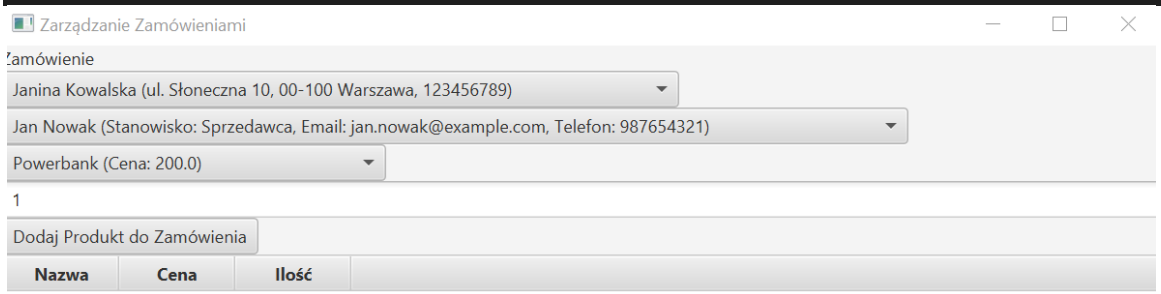
```
CREATE OR REPLACE PROCEDURE anuluj_zamowienie(
    p_zamowienie_id IN NUMBER
)
IS
BEGIN
    DELETE FROM SzczegolyZamowienia WHERE ZamowieniaID = p_zamowienie_id;
    DELETE FROM Zamowienia WHERE ZamowienieID = p_zamowienie_id;
    COMMIT;
END anuluj_zamowienie;
/
```

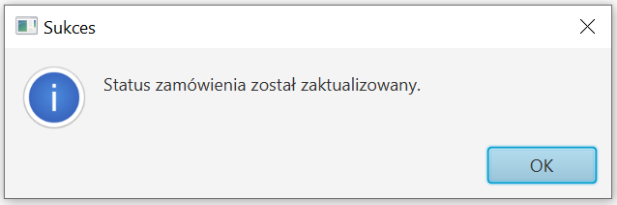


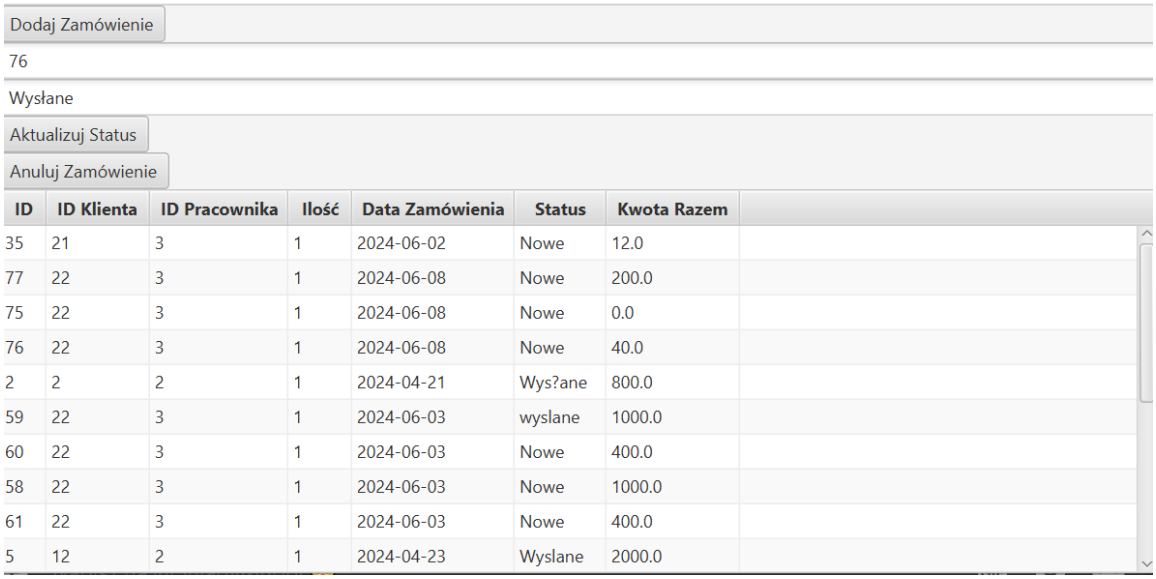
- **Aktualizowanie statusu zamówienia:**
  -funkcja, która aktualizuje status zamówienia. Pobiera id zamówienia i nowy status dla zamówienia. Następnie aktualizuje kolumnę Status dla zamówienia o wcześniej pobranym id.

```
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.

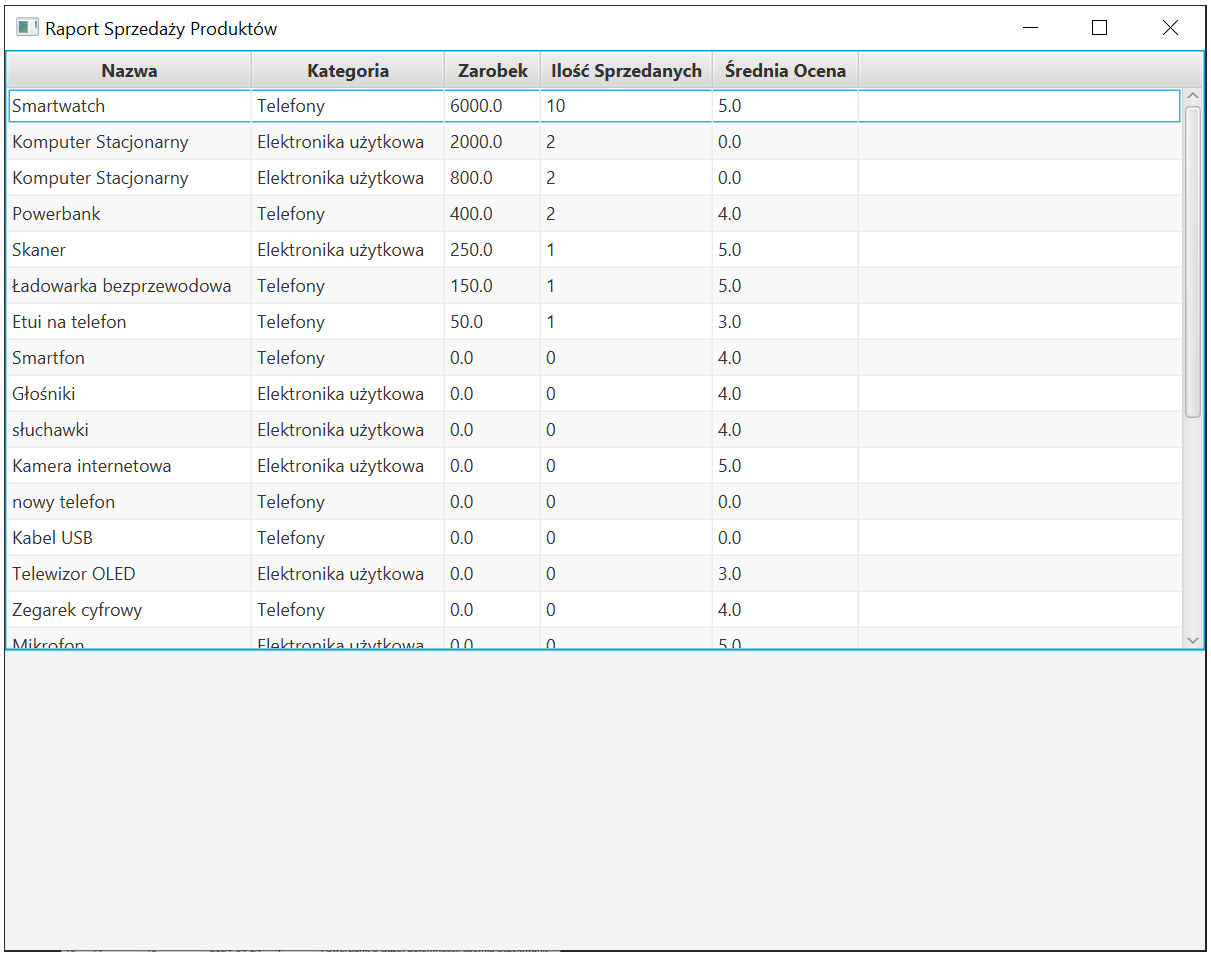
```
CREATE OR REPLACE FUNCTION cena_po_promocji(
    p_produkt_id IN NUMBER
) RETURN NUMBER
IS
    v_cena NUMBER;
    v_procent_obnizki NUMBER;
BEGIN
    SELECT p.CenaJednostkowa, pr.ProcentObnizki
    INTO v_cena, v_procent_obnizki
    FROM Produkty p
    JOIN Promocje pr ON p.PromocjaID = pr.PromocjaID
    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;
/
```



Nazwa	Kategoria	Zarobek	Ilość Sprzedanych	Średnia Ocena
Smartwatch	Telefony	6000.0	10	5.0
Komputer Stacjonarny	Elektronika użytkowa	2000.0	2	0.0
Komputer Stacjonarny	Elektronika użytkowa	800.0	2	0.0
Powerbank	Telefony	400.0	2	4.0
Skaner	Elektronika użytkowa	250.0	1	5.0
Ładowarka bezprzewodowa	Telefony	150.0	1	5.0
Etui na telefon	Telefony	50.0	1	3.0
Smartfon	Telefony	0.0	0	4.0
Głośniki	Elektronika użytkowa	0.0	0	4.0
słuchawki	Elektronika użytkowa	0.0	0	4.0
Kamera internetowa	Elektronika użytkowa	0.0	0	5.0
nowy telefon	Telefony	0.0	0	0.0
Kabel USB	Telefony	0.0	0	0.0
Telewizor OLED	Elektronika użytkowa	0.0	0	3.0
Zegarek cyfrowy	Telefony	0.0	0	4.0
Mikrofon	Elektronika użytkowa	0.0	0	5.0

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.