POLITECHNIKA RZESZOWSKA im. IGNACEGO ŁUKASIEWICZA

Wydział Matematyki i Fizyki Stosowanej

Inżynieria i analiza danych

PROJEKT

Aplikacje Bazodanowe

Komis samochodowy

Daniel Krzysik 166667

Rzeszów 2022

Spis treści

1. Cel projektu 4
2. Baza danych 4
3. Procedury 5
1. DODAJ_KLIENTA 5
2. DODAJ_SAMOCHÓD 6
3. EDYTUJ_KLIENTA 8
4. SREDNIA_CENA_OSTATNIEGO 10
5. SZCZEGOLY_TRANSAKCJI 11
6. USUN_SAMOCHOD 13
7. WYSWIETL_KLIENTOW 14
8. WYSWIETL_SAMOCHODY_O_PRZEBIEGU 15
9. ZATRUDNIJ_SPRZEDAWCE 16
10. ZMIEN_SPRZEDAWCE_SAMOCHODU 16
11. ZWIEKSZPENSJESPRZEDAWCY 18
12. ZWOLNIJSPRZEDAWCE 19
4. Funkcje 20
1. CENA_Z_RABATEM 20
2. CZY_KLIENT_KUPIL_SAMOCHOD 22
3. CZY_SAMOCHOD_DOSTEPNY 23
4. NAJCZESCIEJ_KUPOWANA_MARKA 23
5. NAJCZESTSZE_KOLORY_SAMOCHODOW 24
6. PROCENTOWY_UDZIAL_MARKI 25
7. SREDNIA_CENA_MARKI 27
8. SREDNI_ROCZNIKSAMOCHODOW_DANEJ_MARKI 27
9. TOP_3_NAJLEPSZYCH_SPRZEDAWCOW 29
10. WYSWIETL_MODELE_MARKI 29
11. WYSWIETL_SAMOCHODY_W_CENIE 30
12. WYSZUKAJ_SAMOCHODY 31
5. Pakiety 33
1. PAKIET1 33
2. PAKIET2 33
3. PAKIET3 33
6. Aplikacja 34
1. HOME: 34
2. DASHBOARD 34
3. Dodaj klienta 38

4. Wyświetl samochody o przebiegu ... 47
5. Dodaj samochód ... 50
6. Edytuj klienta ... 56
7. Usuń samochód ... 61
8. Średnia cena ostatniego ... 63
9. Szczegóły ostatniej transakcji ... 65
10. Wyświetl klientów ... 67
11. Zatrudnij sprzedawcę ... 69
12. Zmień sprzedawcę samochodu ... 72
13. Zwiększ pensje sprzedawcy ... 74
14. Zwolnij sprzedawcę ... 75
15. Top3 sprzedawców ... 77
16. Cena po rabacie ... 79
17. Czy klient zakupił samochód ... 81
18. Czy samochód jest dostępny ... 84
19. Najczęściej kupowana marka ... 86
20. Najczęstsze kolory samochodów ... 88
21. Procentowy udział marki ... 89
22. Średnia cena marki ... 91
23. Średni rocznik samochodów danej marki ... 92
24. Wyświetl modele marki ... 95
25. Wyświetl samochody w cenie ... 98
26. Wyszukaj samochody ... 100
7. Podsumowanie ... 105

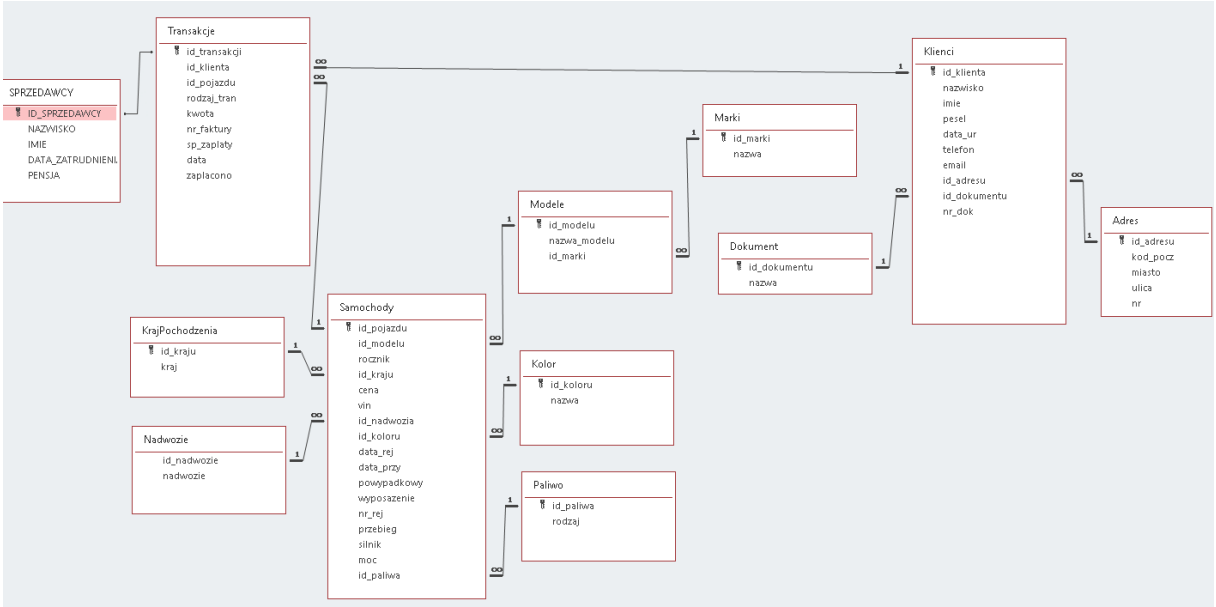
1. Cel projektu

Celem projektu jest stworzenie aplikacji komis samochodowy na platformie Oracle APEX, która pozwoli na zarządzanie ofertą samochodów, obsługę transakcji oraz generowanie raportów i analizowanie danych sprzedaży. Aplikacja ta ma na celu ułatwić i usprawnić procesy związane z działalnością komisu samochodowego, umożliwiając prezentację oferty, sprzedaż samochodów oraz zarządzanie danymi dotyczącymi klientów, sprzedawców i transakcji.

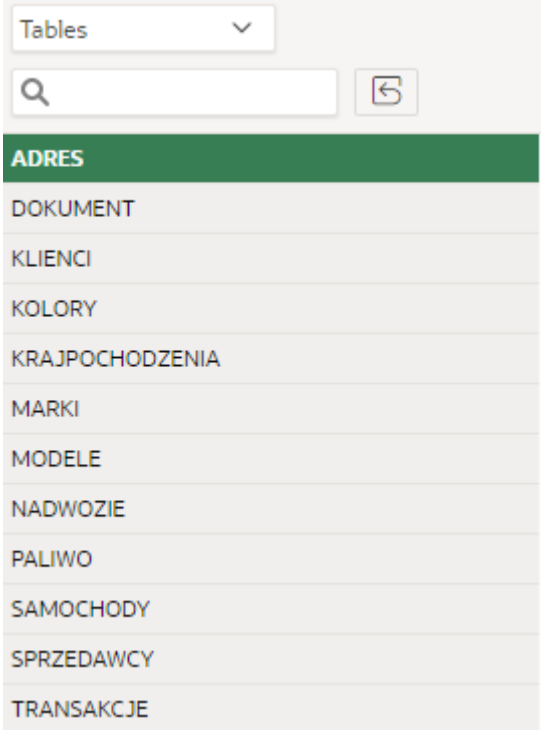
2. Baza danych

Użyta została baza danych z poprzedniego projektu – komis samochodowy.

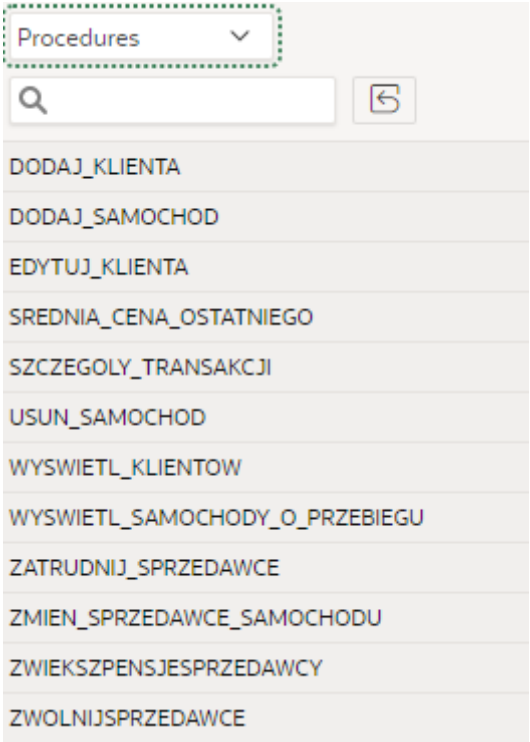
SCHEMAT BAZY



Utworzone tabele:



3. Procedury



1. DODAJ_KLIENTA

Procedura o nazwie "dodaj_klienta" jest używana do dodawania nowego klienta do bazy danych. Procedura przyjmuje kilka parametrów, w tym nazwisko, imię, numer PESEL, datę urodzenia, numer telefonu, adres e-mail, numer dokumentu, nazwę dokumentu i identyfikator adresu. Procedura ma również parametr wyjściowy o nazwie "wynik", który nie jest używany w procedurze.

Procedura w pierwszej kolejności wybiera identyfikator dokumentu z tabeli "dokument" na podstawie jego nazwy. Następnie sprawdza, czy długość numeru PESEL jest równa 11 i czy adres e-mail zawiera ciąg "@gmail.com". Jeśli którykolwiek z tych warunków nie jest spełniony, generowany jest wyjątek i wyświetlany jest odpowiedni komunikat błędu.

Procedura następnie pobiera następną wartość z sekwencji o nazwie "SEQ_KLIENCI" i przypisuje ją do zmiennej "v_id". Następnie procedura wstawia nowy rekord do tabeli "klienci" z wartościami parametrów wejściowych i przypisanym identyfikatorem.

Na końcu procedura posiada dwa obsługi wyjątków dla wyjątków "zly_pesel" i "zly_email", które generują błędy aplikacji z konkretnymi kodami i komunikatami błędów.

KOD:

```oracle_plsql
create or replace PROCEDURE dodaj_klienta(p_nazwisko VARCHAR2, p_imie VARCHAR2,
 p_pesel VARCHAR2, p_data_ur VARCHAR2, p_telefon VARCHAR2, p_email VARCHAR2,
 p_nr_dok VARCHAR2, p_nazwa_dokumentu VARCHAR2, p_id_adresu NUMBER, wynik OUT VARCHAR2)
AS
v_id NUMBER;
v_id_dok NUMBER;
zly_pesel exception;
zly_email exception;
BEGIN
  SELECT ID_DOKUMENTU INTO v_id_dok FROM dokument WHERE nazwa = p_nazwa_dokumentu;

  IF LENGTH(p_pesel) <> 11 THEN raise
    zly_pesel;
  END IF;

  IF INSTR(p_email, '@gmail.com') < 1 THEN raise
    zly_email;
  END IF;

  SELECT SEQ_KLIENCI.NEXTVAL INTO v_id FROM DUAL;

  INSERT INTO klienci (ID_KLIENTA, nazwisko, imie, pesel,
  data_ur, telefon, email, nr_dok, id_dokumentu, id_adresu)
  VALUES (v_id, p_nazwisko, p_imie, p_pesel, TO_DATE(p_data_ur, 'MM/DD/RRRR'),
   p_telefon, p_email, p_nr_dok, v_id_dok, p_id_adresu);
EXCEPTION
  WHEN zly_pesel THEN
    apex_error.add_error(p_message => 'Niepoprawny numer pesel',
            p_display_location => apex_error.c_inline_in_notification);
  WHEN zly_email THEN
    apex_error.add_error(p_message => 'Niepoprawny adres e-mail. Musi zawierać końcówkę @gmail.com',
            p_display_location => apex_error.c_inline_in_notification);
END;
```

2. DODAJ_SAMOCHÓD

Procedura o nazwie "DODAJ_SAMOCHOD" jest używana do dodawania nowego samochodu do bazy danych. Procedura przyjmuje wiele parametrów, w tym rocznik, cenę, numer VIN, daty rejestracji i pierwszej rejestracji, czy samochód jest powypadkowy, numer rejestracyjny, typ silnika, moc, typ nadwozia, rodzaj paliwa, kraj produkcji, model, kolor oraz przebieg. Procedura również ma parametr wyjściowy o nazwie "wynik", który nie jest wykorzystywany w procedurze.

Procedura najpierw sprawdza, czy podany numer VIN już istnieje w tabeli "samochody", a następnie sprawdza czy wszystkie wymagane parametry zostały podane. Jeśli którykolwiek z tych warunków nie jest spełniony, generowany jest wyjątek i wyświetlany jest odpowiedni komunikat błędu.

Procedura pobiera następną wartość z sekwencji o nazwie "SEQ_SAMOCHODY" i przypisuje ją do zmiennej "ID_POJAZDU", następnie wstawia nowy rekord do tabeli "samochody" z wartościami parametrów wejściowych i przypisanym identyfikatorem.

Na końcu procedura posiada trzy obsługi wyjątków dla "zly_rocznik", "zly_vin" i "zly_vin2" wyjątków, które generują błędy aplikacji z konkretnymi kodami i komunikatami błędów.

KOD:

```oracle_plsql
create or replace PROCEDURE DODAJ_SAMOCHOD(
        p_rocznik       IN samochody.ROCZNIK%TYPE,
        p_cena          IN samochody.CENA%TYPE,
        p_vin           IN samochody.VIN%TYPE,
        p_datarej       IN samochody.DATA_REJ%TYPE,
        p_dataprzy      IN samochody.DATA_PRZ%TYPE,
        p_powypad       IN samochody.POWYPADKOWY%TYPE,
        p_nrrej         IN samochody.NR_REJ%TYPE,
        p_silnik        IN samochody.SILNIK%TYPE,
        p_moc           IN samochody.MOC%TYPE,
        p_typnadwozia   IN samochody.ID_NADWOZIE%TYPE,
        p_paliwo        IN samochody.ID_PALIWA%TYPE,
        p_kraj          IN samochody.ID_KRAJU%TYPE,
        p_idmodelu      IN samochody.ID_MODELU%TYPE,
        p_typkoloru     IN samochody.ID_KOLORU%TYPE,
        p_przebieg      IN samochody.PRZEBIEG%TYPE,
        wynik OUT VARCHAR2)
IS
  v_vin_exists NUMBER;
    zly_rocznik exception;
    zly_vin exception;
    zly_vin2 exception;
BEGIN
  SELECT COUNT(*) INTO v_vin_exists FROM samochody WHERE vin = p_vin;
   IF p_rocznik IS NULL OR p_cena IS NULL OR p_vin IS NULL OR p_datarej
    IS NULL OR p_dataprzy IS NULL OR p_nrrej IS NULL OR p_silnik
     IS NULL OR p_moc IS NULL OR p_typnadwozia IS NULL OR p_paliwo
     IS NULL OR p_kraj IS NULL OR p_idmodelu IS NULL OR p_typkoloru
      IS NULL OR p_przebieg
       IS NULL THEN RAISE
     zly_rocznik;
  ELSIF v_vin_exists > 0 THEN RAISE
     zly_vin;
  ELSIF LENGTH(p_vin) <> 11 THEN RAISE
     zly_vin2;
  ELSE
  INSERT INTO samochody ("ID_POJAZDU", "ROCZNIK", "CENA", "VIN",
              "DATA_REJ", "DATA_PRZ", "POWYPADKOWY",
              "NR_REJ", "SILNIK", "MOC", "ID_NADWOZIE", "ID_PALIWA", "ID_KRAJU",
              "ID_MODELU", "ID_KOLORU", "PRZEBIEG")
  VALUES (SEQ_SAMOCHODY.NEXTVAL, p_rocznik, p_cena, p_vin,
  TO_DATE(p_datarej, 'MM/DD/RRRR'), TO_DATE(p_dataprzy, 'MM/DD/RRRR'),
```

```
43    p_powypad, p_nrrej, p_silnik,  p_moc, p_typnadwozia, p_paliwo, p_kraj,
44      p_idmodelu, p_typkoloru, p_przebieg);
45    COMMIT;
46    END IF;
47    EXCEPTION
48    WHEN zly_rocznik THEN
49      apex_error.add_error(p_message => 'Wszystkie wymagane parametry muszą zostać podane',
50              p_display_location => apex_error.c_inline_in_notification);
51    WHEN zly_vin THEN
52      apex_error.add_error(p_message => 'Samochód o tym samym numerze VIN już istnieje',
53              p_display_location => apex_error.c_inline_in_notification);
54    WHEN zly_vin2 THEN
55      apex_error.add_error(p_message => 'Numer VIN musi mieć dokładnie 11 znaków',
56              p_display_location => apex_error.c_inline_in_notification);
57  END;
```

3. EDYTUJ_KLIENTA

Procedura "edytuj_klienta" jest używana do edytowania istniejącego klienta w bazie danych. Procedura przyjmuje jako parametry identyfikator klienta, nazwisko, imię i numer PESEL klienta, oraz trzy parametry wyjściowe, które nie są wykorzystywane w procedurze.

Procedura tworzy kursor "c_klient" z danymi istniejącego klienta o podanym identyfikatorze. Następnie otwiera kursor, pobiera dane do zmiennych lokalnych "l_nazwisko", "l_imie" i "l_PESEL" i zamyka kursor.

Następnie procedura ustawia nowe wartości dla nazwiska, imienia i PESEL, korzystając z funkcji COALESCE, która pozwala na ustawienie nowej wartości tylko jeśli została podana jako parametr wejściowy.

Następnie procedura sprawdza czy podany PESEL jest dłuższy niż 11 znaków, jeśli tak, generowany jest wyjątek "zly_pesel" i wyświetlany jest komunikat błędu.

Jeśli warunek PESEL jest spełniony, procedura aktualizuje rekord w tabeli "klienci" z nowymi wartościami nazwiska, imienia i PESEL, a następnie dokonuje zmiany i zwraca komunikat o zmianie danych.

Na końcu procedura posiada obsługę wyjątku "zly_pesel", który generuje błąd informujący o nieprawidłowym formacie numeru PESEL oraz "NO_DATA_FOUND" który generuje błąd informujący o braku klienta o podanym ID.

```oracle_plsql
create or replace PROCEDURE edytuj_klienta (
    p_id_klienta NUMBER,
    p_nazwisko VARCHAR2,
    p_imie VARCHAR2,
    p_PESEL VARCHAR2,
    wynik OUT VARCHAR2,
    wynik1 OUT VARCHAR2,
    wynik2 OUT VARCHAR2,
    wynik3 OUT VARCHAR2
)
AS
    l_nazwisko_stare VARCHAR2(30);
    l_imie_stare VARCHAR2(30);
    l_PESEL_stare VARCHAR2(11);
    l_nazwisko VARCHAR2(30);
    l_imie VARCHAR2(30);
    l_PESEL VARCHAR2(11);
    zly_pesel exception;

CURSOR c_klient IS
    SELECT nazwisko, imie, PESEL FROM klienci WHERE id_klienta = p_id_klienta;
BEGIN
    SELECT nazwisko, imie, PESEL INTO l_nazwisko_stare, l_imie_stare,
    l_PESEL_stare FROM klienci WHERE id_klienta = p_id_klienta;
    OPEN c_klient;
    FETCH c_klient INTO l_nazwisko, l_imie, l_PESEL;
    CLOSE c_klient;

    l_nazwisko := COALESCE(p_nazwisko, l_nazwisko);
    l_imie := COALESCE(p_imie, l_imie);
    l_PESEL := COALESCE(p_PESEL, l_PESEL);

IF p_PESEL IS NOT NULL AND LENGTH(p_PESEL) <> 11 THEN RAISE
    zly_pesel;
ELSE
    UPDATE klienci
    SET nazwisko = l_nazwisko, imie = l_imie, PESEL = l_PESEL
    WHERE id_klienta = p_id_klienta;
    COMMIT;
```

```
    IF p_nazwisko IS NOT NULL THEN
        wynik1 := ('Zmieniono nazwisko z ' || l_nazwisko_stare || ' na ' || p_nazwisko || '<br>');
    END IF;

    IF p_imie IS NOT NULL THEN
        wynik2 := ('Zmieniono imie z ' || l_imie_stare || ' na ' || p_imie || '<br>');
    END IF;

    IF p_PESEL IS NOT NULL AND LENGTH(p_PESEL) = 11 THEN
        wynik3 := ('Zmieniono PESEL z ' || l_PESEL_stare || ' na ' || p_PESEL || '<br>');
    END IF;
END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
     raise_application_error(-20005, 'Nie znaleziono klienta o takim ID.');
WHEN zly_pesel THEN
    raise_application_error(-20002, 'PESEL musi zawierać 11 znaków!');
END;

```

4. SREDNIA_CENA_OSTATNIEGO

Procedura "srednia_cena_ostatniego" przyjmuje jako parametr nazwę marki samochodu (p_marka) oraz zmienną wynikową, do której zostanie zapisany wynik działania procedury.

Następnie otwierany jest cursor "v_cursor" za pomocą którego pobierane są ceny samochodów marki podanej jako parametr, zapisane w tablicy "tabela_cen_samochodow" . Cursor jest następnie zamykany.

Następnie liczony jest średnią cenę samochodów tej marki, a także pobierany jest ostatni model sprzedanego samochodu tej marki.

Jeżeli nie sprzedano żadnego samochodu tej marki, wynik zostanie ustawiony jako komunikat o braku sprzedanych samochodów tej marki. W przeciwnym razie zostanie zwrócony komunikat z ostatnio sprzedanym modelem oraz średnia cena ostatnio sprzedanego samochodu danej marki.

```
create or replace PROCEDURE srednia_cena_ostatniego
(p_marka VARCHAR2, wynik OUT VARCHAR2)
AS
TYPE tabela_cen_samochodow_typ IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
tabela_cen_samochodow tabela_cen_samochodow_typ;

v_cursor SYS_REFCURSOR;
v_ile NUMBER;
v_suma NUMBER;
v_srednia NUMBER;
v_ostatni_model VARCHAR2(300);
BEGIN

v_suma := 0;
v_ile := 0;

OPEN v_cursor FOR
SELECT s.cena
FROM samochody s
INNER JOIN modele m ON s.id_modelu = m.id_modelu
INNER JOIN marki ma ON m.id_marki = ma.id_marki
WHERE ma.nazwa = p_marka;
```

```
FETCH v_cursor BULK COLLECT INTO tabela_cen_samochodow;
CLOSE v_cursor;

v_ile := tabela_cen_samochodow.COUNT;

IF v_ile = 0 THEN
    wynik := ('Nie sprzedano żadnego samochodu marki: ' || p_marka);
ELSE
    v_ile := 0;
    FOR i IN 1..tabela_cen_samochodow.COUNT LOOP
    v_suma := v_suma + tabela_cen_samochodow(i);
    v_ile := v_ile + 1;
    END LOOP;

    v_srednia := v_suma / v_ile;

    OPEN v_cursor FOR
    SELECT m.nazwa_modelu
    FROM samochody s
    INNER JOIN modele m ON s.id_modelu = m.id_modelu
    INNER JOIN marki ma ON m.id_marki = ma.id_marki
    WHERE ma.nazwa = p_marka
    AND ROWNUM = 1
    ORDER BY DATA_PRZ DESC;

    FETCH v_cursor INTO v_ostatni_model;
    CLOSE v_cursor;
        wynik := ('Ostatnio sprzedany samochód to ' || p_marka
         || ' ' || v_ostatni_model || '. Średnia cena samochodów
          tej marki wynosi ' || round(v_srednia,2) || 'zł.' );
END IF;
END;
```

5. SZCZEGOLY_TRANSAKCJI

Procedura szczegoly_transakcji ma na celu wyświetlenie szczegółów ostatniej transakcji.

Aby to zrobić, najpierw tworzy ona kursor v_cursor, za pomocą którego zapytanie z bazy danych pobiera dane dotyczące ostatniej transakcji typu "Kupno" lub "Sprzedaz", posortowane według daty w kolejności malejącej i zawierające informacje dotyczące transakcji, pojazdu, klienta i sprzedawcy. Następnie dane te są przypisywane do zmiennych lokalnych i są formatowane w celu utworzenia czytelnego i przyjaznego dla użytkownika wyniku, który jest zwracany jako wynik.

```
1  create or replace PROCEDURE szczegoly_transakcji (wynik OUT VARCHAR2)
2  AS
3  v_cursor SYS_REFCURSOR;
4  v_id_transakcji NUMBER;
5  v_rodzaj_trans VARCHAR2(20);
6  v_kwota NUMBER;
7  v_data_trans DATE;
8  v_id_pojazdu NUMBER;
9  v_id_klienta NUMBER;
10 v_id_sprzedawcy NUMBER;
11 v_nazwisko_klienta VARCHAR2(50);
12 v_imie_klienta VARCHAR2(50);
13 v_nazwisko_sprzedawcy VARCHAR2(50);
14 v_imie_sprzedawcy VARCHAR2(50);
15 v_rocznik NUMBER;
16 v_cena NUMBER;
17 v_marka VARCHAR2(30);
18 v_model VARCHAR2(30);
19 v_kraj_pochodzenia VARCHAR2(30);
20 v_nadwozie VARCHAR2(30);
21 v_paliwo VARCHAR2(30);
22 v_kolor VARCHAR2(30);
23 BEGIN
24 OPEN v_cursor FOR
25 SELECT t.id_transakcji, t.rodzaj_tran, t.kwota, t.data, s.id_pojazdu, s.rocznik, s.cena,
26 k.id_klienta, k.nazwisko nazwisko_klienta, k.imie imie_klienta,
27 sp.id_sprzedawcy, sp.nazwisko nazwisko_sprzedawcy, sp.imie imie_sprzedawcy,
28 m.nazwa marka, mo.nazwa_modelu model, kp.kraj, n.nadwozie, ko.nazwa, p.rodzaj
29 FROM transakcje t
30 INNER JOIN samochody s ON t.id_pojazdu = s.id_pojazdu
31 INNER JOIN klienci k ON t.id_klienta = k.id_klienta
32 INNER JOIN sprzedawcy sp ON t.id_sprzedawcy = sp.id_sprzedawcy
33 INNER JOIN modele mo ON s.id_modelu = mo.id_modelu
34 INNER JOIN marki m ON mo.id_marki = m.id_marki
35 INNER JOIN krajpochodzenia kp ON s.id_kraju = kp.id_kraju
36 INNER JOIN kolory ko ON s.id_koloru = ko.id_koloru
37 INNER JOIN paliwo p ON s.id_paliwa = p.id_paliwa
38 INNER JOIN nadwozie n ON s.id_nadwozie = n.id_nadwozie
39 WHERE t.rodzaj_tran = 'Kupno' OR t.rodzaj_tran = 'Sprzedaz'
40 AND ROWNUM = 1
41 ORDER BY t.DATA DESC;
```

```oracle_plsql

FETCH v_cursor INTO v_id_transakcji, v_rodzaj_trans, v_kwota,
 v_data_trans, v_id_pojazdu, v_rocznik, v_cena, v_id_klienta,
 v_nazwisko_klienta, v_imie_klienta, v_id_sprzedawcy,
 v_nazwisko_sprzedawcy, v_imie_sprzedawcy, v_marka, v_model,
 v_kraj_pochodzenia, v_nadwozie, v_kolor, v_paliwo;

 wynik := ('Szczegóły ostatniej transakcji:' || '<br>' ||
'ID transakcji: ' || v_id_transakcji || '<br>' ||
'Rodzaj transakcji: ' || v_rodzaj_trans || '<br>' ||
'Kwota: ' || v_kwota || ' zł' || '<br>' ||
'Data transakcji: ' || TO_CHAR(v_data_trans, 'MM/DD/RRRR') || '<br>' ||
'ID pojazdu: ' || v_id_pojazdu || '<br>' ||
'Marka: ' || v_marka || '<br>' ||
'Model: ' || v_model || '<br>' ||
'Rocznik: ' || v_rocznik || '<br>' ||
'Cena: ' || v_cena || ' zł' || '<br>' ||
'Marka: ' || v_kraj_pochodzenia || '<br>' ||
'Model: ' || v_nadwozie || '<br>' ||
'Marka: ' || v_paliwo || '<br>' ||
'Model: ' || v_kolor || '<br>' ||
'ID klienta: ' || v_id_klienta || '<br>' ||
'Nazwisko klienta: ' || v_nazwisko_klienta || '<br>' ||
' Imię klienta: ' || v_imie_klienta || '<br>' ||
'ID sprzedawcy: ' || v_id_sprzedawcy || '<br>' ||
'Nazwisko sprzedawcy: ' || v_nazwisko_sprzedawcy || '<br>' ||
' Imię sprzedawcy: ' || v_imie_sprzedawcy || '<br>');

CLOSE v_cursor;

END;
```

6. USUN_SAMOCHOD

Procedura "usun_samochod" usuwa samochód o podanym ID z bazy danych, a także zwraca informację o tym jaki samochód został usunięty (marka, model, ID). Jeśli nie znaleziono samochodu o podanym ID, procedura zgłasza błąd.

```oracle
create or replace PROCEDURE usun_samochod (
p_id_samochodu NUMBER,
    wynik OUT VARCHAR2
)
AS
marka VARCHAR2(100);
model VARCHAR2(100);
BEGIN
SELECT ma.nazwa as marka, m.nazwa_modelu as model
INTO marka, model
FROM samochody s
INNER JOIN modele m ON s.id_modelu = m.id_modelu
INNER JOIN marki ma ON m.id_marki = ma.id_marki
WHERE s.id_pojazdu = p_id_samochodu;

DELETE FROM samochody
WHERE id_pojazdu = p_id_samochodu;
COMMIT;

wynik := ('Usunięto samochod marki ' || marka || ' model '
|| model || ' o ID ' || p_id_samochodu || ' z bazy danych.');
EXCEPTION
WHEN NO_DATA_FOUND THEN
raise_application_error(-20005, 'Nie znaleziono samochodu o takim ID.');
END;
```

7. WYSWIETL_KLIENTOW

Procedura "wyswietl_klientow" tworzy kursor "c_klienci" który pobiera dane z tabeli "klienci" w kolejności alfabetycznej po nazwisku. Następnie iteruje przez każdy rekord z kursoru i zapisuje do zmiennej "wynik1" nazwisko, imię i wiek klienta w formacie "Nazwisko Imie Wiek lat". Wynik jest zwracany jako wynik1 i wynik.

```
create or replace PROCEDURE wyswietl_klientow
(wynik OUT VARCHAR2, wynik1 OUT VARCHAR2) AS
    CURSOR c_klienci IS
        SELECT nazwisko, imie,
        TRUNC(TO_NUMBER(SYSDATE- TO_DATE(data_ur))/365.25) AS Wiek
        FROM klienci
        ORDER BY nazwisko;
BEGIN
    wynik := ('Lista klientów:');
    wynik1 := '';
    FOR r_klienci IN c_klienci LOOP
        wynik1 := (wynik1 || r_klienci.nazwisko || ' '
        || r_klienci.imie || ' ' || r_klienci.Wiek || ' lat' || '<br>');
    END LOOP;
END;
```

8. WYSWIETL_SAMOCHODY_O_PRZEBIEGU

Procedura "wyswietl_samochody_o_przebiegu" przyjmuje 2 parametry wejściowe - p_przebieg_od i p_przebieg_do, które reprezentują minimalny i maksymalny przebieg samochodów, które mają zostać wyświetlone. Następnie tworzy kursor "c_samochody", który zawiera dane z tabeli "samochody", łączy ją z tabelami "modele" i "marki" i filtruje wyniki wg podanego zakresu przebiegu. Kursor sortuje wyniki po marce i modelu.

Procedura używa pakietu HTP, aby wyświetlić tekst "Lista samochodów o przebiegu pomiędzy x km a y km:" oraz dane poszczególnych samochodów, zawarte w kursorze "c_samochody", w formie "marka model, rocznik x, przebieg y km, cena z zł". Jeśli nie znajdzie żadnych pasujących rekordów, zostanie wyświetlony komunikat "Nie znaleziono samochodów spełniających podane kryteria."

```
create or replace PROCEDURE wyswietl_samochody_o_przebiegu
(p_przebieg_od NUMBER, p_przebieg_do NUMBER) AS
CURSOR c_samochody IS
SELECT ma.nazwa as marka, m.nazwa_modelu as model,
 s.rocznik, s.przebieg, s.cena
FROM samochody s
INNER JOIN modele m ON s.id_modelu = m.id_modelu
INNER JOIN marki ma ON m.id_marki = ma.id_marki
WHERE s.przebieg BETWEEN p_przebieg_od AND p_przebieg_do
ORDER BY marka, model;
BEGIN
HTP.P('Lista samochodów o przebiegu pomiędzy ' ||
p_przebieg_od || 'km a ' || p_przebieg_do || 'km:' || '<br>');
FOR r_samochody IN c_samochody LOOP
HTP.P(r_samochody.marka || ' ' || r_samochody.model ||
', rocznik ' || r_samochody.rocznik || ', przebieg ' ||
r_samochody.przebieg || 'km, cena ' || r_samochody.cena ||
' zł' || '<br>');
END LOOP;
```

```
EXCEPTION
WHEN NO_DATA_FOUND THEN
apex_error.add_error(p_message => 'Nie znaleziono samochodów spełniających podane kryteria.',
            p_display_location => apex_error.c_inline_in_notification);
END;
```

9. ZATRUDNIJ_SPRZEDAWCE

Procedura "zatrudnij_Sprzedawce" jest przeznaczona do zatrudniania nowych sprzedawców w bazie danych. Przyjmuje ona trzy parametry wejściowe: imię, nazwisko i pensję oraz jeden parametr wyjściowy: wynik.

Wewnątrz procedury tworzony jest nowy rekord w tabeli "SPRZEDAWCY" z podanymi danymi. Przed dodaniem rekordu sprawdzana jest czy pensja jest większa niż 1490. Jeśli nie jest, procedura rzuca wyjątek "NO_DATA_FOUND", który przechwytuje obsługa wyjątków i podaje komunikat o błędnej pensji.

Jeśli wszystko przebiega prawidłowo, wynik zwraca komunikat o poprawnym zatrudnieniu sprzedawcy.

```
create or replace PROCEDURE zatrudnij_Sprzedawce (
    p_imie VARCHAR2,
    p_nazwisko VARCHAR2,
    p_pensja NUMBER,
    wynik OUT VARCHAR2
)
IS
v_id NUMBER;
v_datazatrudnienia DATE;
BEGIN
v_datazatrudnienia := SYSDATE;
IF p_pensja <= 1490 THEN
    RAISE NO_DATA_FOUND;
END IF;
    SELECT SEQ_SPRZEDAWCY.NEXTVAL INTO v_id FROM DUAL;
INSERT INTO "SPRZEDAWCY" ("ID_SPRZEDAWCY", "NAZWISKO", "IMIE", "DATA_ZATRUDNIENIA", "PENSJA")
    VALUES(v_id, p_imie, p_nazwisko, v_datazatrudnienia, p_pensja);
    wynik := ('Zatrudniono sprzedawce ' || p_nazwisko);
EXCEPTION
WHEN NO_DATA_FOUND THEN
raise_application_error(-20005, 'Pensja musi być minimalna.');
END;
```

10. ZMIEN_SPRZEDAWCE_SAMOCHODU

Procedura "zmien_sprzedawce_samochodu" pozwala na zmianę sprzedawcy danego samochodu. Przyjmuje ona trzy parametry wejściowe: ID samochodu, ID nowego sprzedawcy oraz powód zmiany. Procedura sprawdza czy podany samochód oraz sprzedawca istnieją w bazie danych. Jeśli tak, to dokonuje zmiany sprzedawcy w tabeli "transakcje" i zwraca informację o tym jaki sprzedawca został zmieniony na jakiego oraz jaki był powód zmiany. Jeśli nie znajdzie samochodu lub sprzedawcy, zwraca odpowiedni komunikat o błędzie.

```oracle_plsql
create or replace PROCEDURE zmien_sprzedawce_samochodu
(p_id_samochodu NUMBER, p_id_nowego_sprzedawcy NUMBER,
powod VARCHAR2, wynik OUT VARCHAR2)
AS
    v_stary_sprzedawca SPRZEDAWCY%ROWTYPE;
    v_nowy_sprzedawca SPRZEDAWCY%ROWTYPE;
    v_czy_istnieje_sprzedawca NUMBER;
    v_czy_istnieje_samochod NUMBER;
    v_result VARCHAR2(1000);
    brak_samochodu exception;
    brak_sprzedawcy exception;
BEGIN
    SELECT COUNT(*) INTO v_czy_istnieje_sprzedawca
    FROM sprzedawcy
    WHERE id_sprzedawcy = p_id_nowego_sprzedawcy;

    SELECT COUNT(*) INTO v_czy_istnieje_samochod
    FROM transakcje
    WHERE id_pojazdu = p_id_samochodu;

    IF v_czy_istnieje_samochod = 0 THEN RAISE
        brak_samochodu;
```

```oracle_plsql
  ELSIF v_czy_istnieje_sprzedawca = 0 THEN RAISE
      brak_sprzedawcy;
  ELSE
      SELECT * INTO v_stary_sprzedawca
      FROM sprzedawcy
      WHERE id_sprzedawcy = (SELECT id_sprzedawcy
       FROM transakcje WHERE id_pojazdu = p_id_samochodu);

      UPDATE transakcje
      SET id_sprzedawcy = p_id_nowego_sprzedawcy
      WHERE id_pojazdu = p_id_samochodu;

      SELECT * INTO v_nowy_sprzedawca
      FROM sprzedawcy
      WHERE id_sprzedawcy = p_id_nowego_sprzedawcy;

      wynik := ('Zmieniono sprzedawcę z ' || v_stary_sprzedawca.nazwisko
      || ' ' || v_stary_sprzedawca.imie || ' na ' || v_nowy_sprzedawca.nazwisko
      || ' ' || v_nowy_sprzedawca.imie || '. Powód: ' || powod);
  END IF;
```

```
EXCEPTION
WHEN brak_samochodu THEN
    raise_application_error(-20005, 'W bazie danych nie ma takiego samochodu.');
WHEN brak_sprzedawcy THEN
    raise_application_error(-20002, 'W bazie danych nie ma takiego sprzedawcy.');
END;

```

11. ZWIEKSZPENSJESPRZEDAWCY

Procedura "zwiekszPensjeSprzedawcy" przyjmuje jako parametr identyfikator pracownika i zwiększa jego pensję o odpowiedni procent (10% lub 15%, w zależności od obecnej pensji). W przypadku braku pracownika o podanym identyfikatorze zgłasza błąd. Wynik zawiera informacje o zmienionej pensji.

```
create or replace PROCEDURE zwiekszPensjeSprzedawcy (
p_id_sprzedawcy NUMBER,
wynik OUT VARCHAR2
)
AS
brak exception;
    CURSOR sprzedawcy_podwyzka IS
      SELECT ID_SPRZEDAWCY,
             PENSJA,
             NAZWISKO
      FROM   SPRZEDAWCY
      WHERE  ID_SPRZEDAWCY = p_id_sprzedawcy
      FOR UPDATE;
    podwyzka NUMBER;
    updated_rows INTEGER := 0;
    v_stara_pensja NUMBER;
    v_nowa_pensja NUMBER;
    v_nazwisko VARCHAR2(100);
BEGIN
    FOR sprzedawcy_rec IN sprzedawcy_podwyzka LOOP
        v_nazwisko := sprzedawcy_rec.nazwisko;
        v_stara_pensja := sprzedawcy_rec.pensja;
```

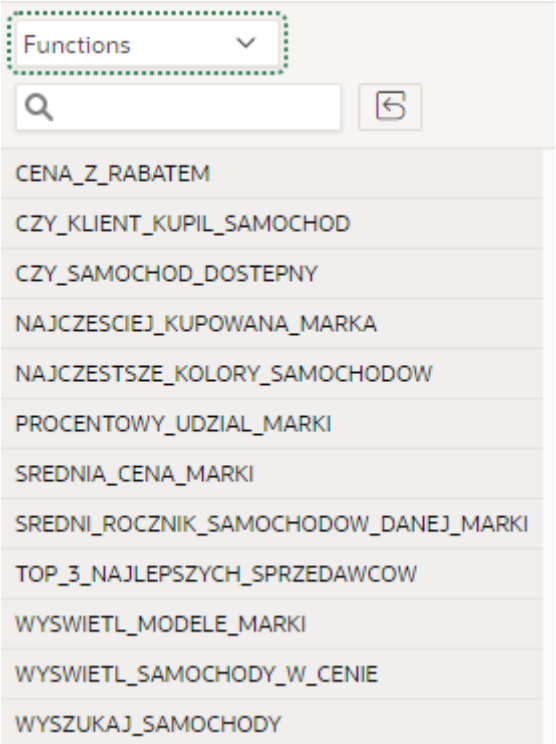
```
        IF sprzedawcy_rec.pensja < 5000 THEN
          podwyzka := .15;
        ELSE
          podwyzka := .10;
        END IF;

        UPDATE SPRZEDAWCY
        SET pensja = pensja + pensja * podwyzka
        WHERE CURRENT OF sprzedawcy_podwyzka;
        updated_rows := updated_rows + 1;
    END LOOP;
    IF updated_rows = 0 THEN RAISE
      brak;
    END IF;
    SELECT pensja INTO v_nowa_pensja FROM sprzedawcy
    WHERE id_sprzedawcy = p_id_sprzedawcy;
    wynik := 'Zmieniono pensję pracownikowi ' || v_nazwisko
    || ' z ' || v_stara_pensja || ' na ' || v_nowa_pensja;
EXCEPTION
WHEN brak THEN
    raise_application_error(-20005, 'Nie znaleziono pracownika o takim ID.');
END;
```

12. ZWOLNIJSPRZEDAWCE

Procedura "zwolnijSprzedawce" usuwa sprzedawcę o podanym ID z bazy danych, przy czym przed usunięciem sprawdza czy taki sprzedawca istnieje. Jeśli sprzedawca istnieje, zostanie usunięty z bazy danych, a wynik zostanie ustawiony na komunikat o usunięciu sprzedawcy. Jeśli sprzedawca nie istnieje, zostanie zgłoszony błąd o braku takiego sprzedawcy.

```oracle_plsql
create or replace PROCEDURE zwolnijSprzedawce (
    p_id NUMBER,
    p_powod VARCHAR2,
    wynik OUT VARCHAR2)
IS
    imie VARCHAR2(50);
    nazwisko VARCHAR2(50);
    dataDzis DATE;
BEGIN
    SELECT imie, nazwisko, SYSDATE
    INTO imie, nazwisko, dataDzis
    FROM "SPRZEDAWCY"
    WHERE "ID_SPRZEDAWCY" = p_id;
DELETE FROM "SPRZEDAWCY" WHERE "ID_SPRZEDAWCY" = p_id;
wynik := ('Zwolniono ' || nazwisko || ' ' || imie ||
 ' dnia: ' || dataDzis || ' z powodu ' || p_powod);
EXCEPTION
WHEN NO_DATA_FOUND THEN
     raise_application_error(-20005, 'Nie można usunąć
     sprzedawcy o podanym ID. Sprawdź, czy taki sprzedawca istnieje
      i czy nie są do niego przypisane żadne transakcje.');
END;
```

4. Funkcje



1. CENA_Z_RABATEM

Funkcja "cena_z_rabatem" przyjmuje jako argumenty ID samochodu oraz wartość rabatu, a następnie oblicza i zwraca nową cenę samochodu po uwzględnieniu rabatu. Jeśli podany ID samochodu nie

istnieje, zostanie zgłoszony błąd. Jeśli podany rabat jest poza zakresem 1-90%, również zostanie zgłoszony błąd. Funkcja aktualizuje również cenę samochodu w bazie danych.

```
1  create or replace FUNCTION cena_z_rabatem (
2      p_id_samochodu NUMBER,
3      p_rabat NUMBER
4  ) RETURN VARCHAR2
5  AS
6      cena_po_rabacie NUMBER;
7      stara_cena NUMBER;
8      wynik VARCHAR2(200);
9  BEGIN
10     SELECT cena, cena - cena * p_rabat / 100
11     INTO stara_cena, cena_po_rabacie
12     FROM samochody
13     WHERE ID_POJAZDU = p_id_samochodu;
14
15 IF stara_cena IS NULL THEN
16    RAISE NO_DATA_FOUND;
17 END IF;
18
19 IF p_rabat < 1 OR p_rabat > 90 THEN
20     RAISE VALUE_ERROR;
21 END IF;
22
```

```
wynik := ('Samochod o ID ' || p_id_samochodu ||
' kosztował ' || stara_cena || ' zł. Rabat w wysokości '
|| p_rabat || '% obniżył cenę do ' || cena_po_rabacie || ' zł');
UPDATE samochody
SET cena = cena_po_rabacie
WHERE ID_POJAZDU = p_id_samochodu;

RETURN wynik;

EXCEPTION
WHEN NO_DATA_FOUND THEN
raise_application_error(-20005, 'Nie znaleziono samochodu o takim ID.');
WHEN VALUE_ERROR THEN
raise_application_error(-20005, 'Rabat musi być w zakresie 1-90%.');
END;
```

2. CZY_KLIENT_KUPIL_SAMOCHOD

Funkcja "czy_klient_kupil_samochod" przyjmuje jeden argument - ID klienta i zwraca informację o tym, czy klient o podanym ID zakupił jakiś samochód oraz ile samochodów zakupił i jaka była suma tych zakupów. Jeśli klient o podanym ID nie istnieje, funkcja zwraca odpowiedni komunikat.

```
create or replace FUNCTION czy_klient_kupil_samochod (p_klient_id NUMBER)
RETURN VARCHAR2
AS
    v_czy_kupil NUMBER;
    v_result VARCHAR2(4000);
    v_czy_istnieje NUMBER;
    v_klient klienci%ROWTYPE;
    v_liczba NUMBER;
    v_sum_cena NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_czy_istnieje
    FROM klienci
    WHERE id_klienta = p_klient_id;

    IF v_czy_istnieje = 0 THEN
        v_result := ('W bazie danych nie ma takiego klienta.');
        RETURN v_result;
    ELSE
        SELECT * INTO v_klient
        FROM klienci
        WHERE id_klienta = p_klient_id;

```

```
        SELECT COUNT(*) INTO v_czy_kupil
        FROM transakcje
        WHERE id_klienta = p_klient_id;

        IF v_czy_kupil > 0 THEN
            SELECT COUNT(*) INTO v_liczba FROM transakcje
            WHERE id_klienta = p_klient_id;
            SELECT SUM(kwota) INTO v_sum_cena FROM transakcje
            WHERE id_klienta = p_klient_id;
            v_result := ('Klient ' || v_klient.imie || ' ' ||
            v_klient.nazwisko || ' zakupił ' || v_liczba ||
            ' samochodów, za które zapłacił łącznie: ' || v_sum_cena || 'zł.');
        ELSE
            v_result := ('Klient ' || v_klient.imie || ' ' ||
            v_klient.nazwisko || ' nie zakupił żadnego samochodu.');
        END IF;
    END IF;
RETURN v_result;
END;
```

3. CZY_SAMOCHOD_DOSTEPNY

Funkcja "czy_samochod_dostepny" przyjmuje jako parametr ID samochodu i zwraca informację czy jest on dostępny (nie został jeszcze sprzedany) czy niedostępny (został już sprzedany). Funkcja wykonuje zapytanie do bazy danych, aby sprawdzić czy dany samochód znajduje się w tabeli transakcji, gdzie rodzaj transakcji jest "Kupno" i zwraca informację o dostępności tego samochodu.

```
create or replace FUNCTION czy_samochod_dostepny(p_id_samochodu NUMBER) RETURN VARCHAR2
AS
    v_dostepnosc VARCHAR2(30);
    v_rocznik NUMBER;
    v_marka VARCHAR2(30);
    v_model VARCHAR2(30);
    wynik VARCHAR2(2000);
BEGIN
    SELECT s.rocznik, ma.nazwa, m.nazwa_modelu INTO v_rocznik, v_marka, v_model
    FROM samochody s INNER JOIN modele m ON s.id_modelu = m.id_modelu
    INNER JOIN marki ma ON m.id_marki = ma.id_marki
    WHERE ID_POJAZDU = p_id_samochodu;

    SELECT CASE WHEN COUNT(*) = 0 THEN 'dostępny' ELSE 'niedostępny' END
    INTO v_dostepnosc
    FROM transakcje
WHERE ID_POJAZDU = p_id_samochodu AND rodzaj_tran = 'Kupno';
wynik := ('Samochód: ' || v_marka || ' ' || v_model ||
' z roku ' || v_rocznik || ' jest ' || v_dostepnosc);
RETURN wynik;
END;
```

4. NAJCZESCIEJ_KUPOWANA_MARKA

Funkcja "najczesciej_kupowana_marka" zwraca informację o najczęściej kupowanej marce samochodów w bazie danych. Funkcja korzysta z kursorów, aby przeszukać tabelę "samochody" i policzyć ilość kupionych modeli każdej marki. Następnie porównuje ilości i zwraca nazwę marki z największą ilością modeli kupionych. Jeśli nie ma samochodów w bazie danych zwraca odpowiedni komunikat.

```oracle_plsql
create or replace FUNCTION najczesciej_kupowana_marka
RETURN VARCHAR2
AS
    marka_samochodu VARCHAR2(30);
    v_ilosc NUMBER;
    v_max_ilosc NUMBER;
CURSOR c_samochody IS
    SELECT ma.nazwa marka_samochodu, COUNT(*) ilosc
    FROM samochody s INNER JOIN modele m ON s.id_modelu = m.id_modelu
    INNER JOIN marki ma ON m.id_marki = ma.id_marki
    GROUP BY ma.nazwa;
BEGIN
    v_max_ilosc := 0;
    marka_samochodu := NULL;
    FOR r_samochody IN c_samochody LOOP
        v_ilosc := r_samochody.ilosc;
        IF v_ilosc > v_max_ilosc THEN
            v_max_ilosc := v_ilosc;
            marka_samochodu := r_samochody.marka_samochodu;
        END IF;
    END LOOP;
    RETURN 'Najczęściej kupowaną marką przez naszych klientów jest: ' ||
    marka_samochodu || ', kupiono ' || v_max_ilosc || ' modeli.';

EXCEPTION
WHEN NO_DATA_FOUND THEN
RETURN 'Brak samochodów w bazie danych';
END;
```

5. NAJCZESTSZE_KOLORY_SAMOCHODOW

Funkcja "najczestsze_kolory_samochodow" zwraca informację o najczęściej występujących kolorach samochodów w bazie danych. Jest to realizowane przez przeszukanie tabeli "kolory" i "samochody" i agregowanie danych o ilości wystąpień danego koloru oraz przedstawienie ich w formie tekstu. Jeśli w bazie danych nie ma żadnych danych dotyczących kolorów samochodów, zostanie rzucony wyjątek.

```oracle_plsql
create or replace FUNCTION najczestsze_kolory_samochodow RETURN VARCHAR2
AS
v_result VARCHAR2(4000);
CURSOR c_kolory IS
SELECT k.nazwa, COUNT(s.id_koloru) liczba
FROM kolory k
INNER JOIN samochody s ON s.id_koloru = k.id_koloru
GROUP BY k.nazwa
ORDER BY liczba DESC;
BEGIN
v_result := ('Najczęściej występujące kolory samochodów: ');
FOR r_kolory IN c_kolory LOOP
v_result := (v_result || r_kolory.nazwa || ' (' || r_kolory.liczba || ' szt.), ');
END LOOP;
RETURN SUBSTR(v_result, 1, LENGTH(v_result)-2);
EXCEPTION
WHEN NO_DATA_FOUND THEN
raise_application_error(-20005, 'Brak danych dotyczących kolorów samochodów.');
END;
```

6. PROCENTOWY_UDZIAL_MARKI

Funkcja "procentowy_udzial_marki" oblicza procentowy udział sprzedaży określonej marki samochodów w salonie samochodowym, na podstawie nazwy marki podanej jako parametr wejściowy (p_marka). Funkcja posiada dwie obsługi wyjątków, "nic" i "brak", które obsługują sytuacje, gdy parametr wejściowy jest brak lub gdy marka nie zostanie odnaleziona w rekordach salonu. Jeśli dane wejściowe są prawidłowe, funkcja oblicza procentowy udział sprzedaży marki w salonie, porównując sumę cen sprzedanych samochodów tej marki z całkowitą sumą cen sprzedanych samochodów.

```oracle_plsql
create or replace FUNCTION procentowy_udzial_marki
(p_marka VARCHAR2 DEFAULT NULL) RETURN VARCHAR2
AS
    v_count NUMBER;
    v_suma NUMBER;
    v_udzial NUMBER;
    v_suma_cen NUMBER;
    v_result VARCHAR2(4000);
    nic exception;
    brak exception;
BEGIN
    IF p_marka IS NULL THEN RAISE
    nic;
    ELSE
    SELECT COUNT(*) INTO v_count
    FROM marki
    WHERE nazwa = p_marka;

    IF v_count = 0 THEN RAISE
        brak;
    ELSE
        SELECT SUM(cena) INTO v_suma FROM samochody WHERE
        id_modelu IN (SELECT id_modelu FROM modele WHERE
        ID_MARKI IN (SELECT ID_MARKI FROM marki WHERE nazwa = p_marka));
        SELECT SUM(cena) INTO v_suma_cen FROM samochody;

        IF v_suma_cen = 0 THEN
            v_udzial := 0;
        ELSE
            v_udzial := v_suma / v_suma_cen * 100;
        END IF;
        v_result := ('Procentowy udział tej marki w naszym komisie wynosi: '
        || TO_CHAR(round(v_udzial,2)) || '%.');
        RETURN v_result;
    END IF;
END IF;
EXCEPTION
    WHEN brak THEN
    raise_application_error(-20005, 'Nie znaleziono ' || p_marka || ' w naszym komisie.');
    WHEN nic THEN
    raise_application_error(-20001, 'Podaj markę samochodu!');
END;
```

7. SREDNIA_CENA_MARKI

Funkcja "SREDNIA_CENA_MARKI" oblicza średnią cenę dla określonej marki samochodów na podstawie nazwy marki przekazanej jako parametr wejściowy (p_marka) i zwraca ciąg znaków, który zawiera tę średnią cenę oraz wszystkie modele samochodów tej marki dostępne w bazie danych.

```oracle
create or replace FUNCTION SREDNIA_CENA_MARKI(p_marka VARCHAR2)
RETURN VARCHAR2
AS
    v_suma NUMBER;
    v_liczba NUMBER;
    v_srednia NUMBER;
    v_model VARCHAR2(30);
    v_count NUMBER;
    v_result VARCHAR2(4000);
BEGIN
SELECT SUM(cena), COUNT(*) INTO v_suma, v_liczba
FROM samochody WHERE id_modelu IN (SELECT id_modelu
FROM modele WHERE ID_MARKI IN (SELECT ID_MARKI
FROM marki WHERE NAZWA = p_marka));
IF v_liczba = 0 THEN
    v_srednia := 0;
ELSE
    v_srednia := v_suma / v_liczba;
END IF;
  v_result := ('Średnia cena samochodów o marce ' || p_marka
  || ' wynosi ' || ROUND(v_srednia, 2) || ' zł.' || '<br>');
  v_result := (v_result || ' Modelami tej marki są samochody: ' || '<br>');

    FOR i IN
        (SELECT id_modelu, COUNT(*) AS model_count FROM samochody
        WHERE id_modelu IN (SELECT id_modelu FROM modele
        WHERE ID_MARKI IN (SELECT ID_MARKI FROM marki
        WHERE NAZWA = p_marka)) GROUP BY id_modelu) LOOP
        SELECT NAZWA_MODELU INTO v_model FROM modele
        WHERE id_modelu = i.id_modelu;
            v_result := (v_result || v_model || ' wystąpił ' ||
            i.model_count || ' razy.' || '<br>');
    END LOOP;

    RETURN v_result;
END;
```

8. SREDNI_ROCZNIKSAMOCHODOW_DANEJ_MARKI

Funkcja "sredni_rocznik_samochodow_danej_marki" oblicza średni rocznik samochodów danej marki na podstawie nazwy marki przekazanej jako parametr wejściowy (p_marka) i zwraca wynik jako ciąg znaków. Funkcja ma trzy obsługi wyjątków, "blad", "blad1" i "NO_DATA_FOUND" do obsługi sytuacji, gdy parametr wejściowy jest pusty, podana marka nie istnieje w bazie danych lub brak samochodów danej marki w bazie danych.

```oracle_plsql
create or replace FUNCTION sredni_rocznik_samochodow_danej_marki
(p_marka VARCHAR2) RETURN VARCHAR2
AS
    v_suma NUMBER;
    v_liczba NUMBER;
    v_sredni_wiek NUMBER;
    v_rok NUMBER;
    blad exception;
    blad1 exception;
BEGIN
IF p_marka IS NULL OR p_marka = '' THEN RAISE
    blad;
ELSE
    SELECT SUM(EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM DATA_REJ)),
    COUNT(*)
    INTO v_suma, v_liczba
    FROM samochody s
    INNER JOIN modele m ON s.id_modelu = m.id_modelu
    INNER JOIN marki ma ON m.id_marki = ma.id_marki
    WHERE ma.nazwa = p_marka;

    IF v_liczba = 0 THEN RAISE
        blad1;
    ELSE
        SELECT MAX(EXTRACT(YEAR FROM DATA_REJ))
        INTO v_rok
        FROM samochody s
        INNER JOIN modele m ON s.id_modelu = m.id_modelu
        INNER JOIN marki ma ON m.id_marki = ma.id_marki
        WHERE ma.nazwa = p_marka;

        IF v_liczba = 0 THEN
            v_sredni_wiek := 0;
        ELSE
            v_sredni_wiek := v_suma / v_liczba;
        END IF;

        RETURN ('Średni rocznik samochodów marki ' || p_marka ||
        ' wynosi ' || (v_rok - ROUND(v_sredni_wiek, 0)) || ' rok.');
    END IF;

END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
raise_application_error(-20003, 'Podana marka samochodu nie istnieje w bazie danych.');
WHEN blad THEN
raise_application_error(-20005, 'Nie podano marki samochodu');
WHEN blad1 THEN
raise_application_error(-20002, 'Brak samochodów marki ' || p_marka || ' w bazie danych');
END;
```

9. TOP_3_NAJLEPSZYCH_SPRZEDAWCOW

Funkcja "top_3_najlepszych_sprzedawcow" zwraca top 3 najlepszych sprzedawców w firmie, na podstawie ilości sprzedanych samochodów. Wynik zwracany jest w postaci tekstu zawierającego imię i nazwisko sprzedawcy oraz ilość sprzedanych samochodów.

```oracle
create or replace FUNCTION top_3_najlepszych_sprzedawcow
RETURN VARCHAR2
AS
    v_result VARCHAR2(4000);
    v_top_sprzedawcy SPRZEDAWCY%ROWTYPE;
BEGIN
    v_result := ('Najlepszymi sprzedawcami są: ' || '<br>');
    FOR v_top_sprzedawcy IN (
        SELECT s.nazwisko, s.imie, COUNT(t.id_transakcji) as ilosc_samochodow
        FROM transakcje t
        INNER JOIN sprzedawcy s ON t.id_sprzedawcy = s.id_sprzedawcy
        GROUP BY s.nazwisko, s.imie
        ORDER BY ilosc_samochodow DESC
        FETCH FIRST 3 ROWS ONLY
    ) LOOP
        v_result := (v_result || v_top_sprzedawcy.nazwisko ||
         ' ' || v_top_sprzedawcy.imie || ' sprzedał ' ||
         v_top_sprzedawcy.ilosc_samochodow || ' samochodów.' || '<br>');
    END LOOP;
    RETURN v_result;
END;
```

10. WYSWIETL_MODELE_MARKI

Funkcja "wyswietl_modele_marki" przyjmuje jako parametr nazwę marki samochodu (p_marka) i zwraca listę modeli danej marki, które są dostępne w komisie, wraz z ich rocznikiem i ceną. Jeśli nie podano marki lub podana marka nie istnieje w bazie danych, funkcja zgłasza odpowiedni błąd.

```oracle_plsql
create or replace FUNCTION wyswietl_modele_marki (p_marka VARCHAR2)
RETURN VARCHAR2
AS
    v_count NUMBER;
    l_ile NUMBER;
    l_modele VARCHAR2(4000);
    v_result VARCHAR2(4000);
    blad exception;
    blad1 exception;
CURSOR c_modele IS
    SELECT m.nazwa_modelu, s.rocznik, s.cena
    FROM samochody s INNER JOIN modele m ON s.id_modelu = m.id_modelu
    INNER JOIN marki ma ON m.id_marki = ma.id_marki
    WHERE ma.nazwa = p_marka;
BEGIN
    IF p_marka IS NULL THEN RAISE
    blad;
    ELSE
    SELECT COUNT(*) INTO v_count
    FROM marki
    WHERE nazwa = p_marka;

    IF v_count = 0 THEN RAISE
        blad1;
    ELSE
    SELECT COUNT(*) INTO l_ile FROM samochody s INNER JOIN
    modele m ON s.id_modelu = m.id_modelu INNER JOIN marki ma
    ON m.id_marki = ma.id_marki WHERE ma.nazwa = p_marka;
    l_modele := ('Samochodów marki ' || p_marka || ' jest: ' ||
     l_ile || '<br>' || 'Lista modelów:' || '<br>');
    FOR model IN c_modele LOOP
        l_modele := (l_modele || model.nazwa_modelu || ' z roku: ' ||
        model.rocznik || ' w cenie: ' || model.cena || 'zł. ' || '<br>');
    END LOOP;
    RETURN l_modele;
    END IF;
    END IF;
    EXCEPTION
    WHEN blad THEN
    raise_application_error(-20005, 'Podaj markę samochodu!');
    WHEN blad1 THEN
    raise_application_error(-20003, 'Nie znaleziono ' || p_marka || ' w naszym komisie.');
END;
```

11. WYSWIETL_SAMOCHODY_W_CENIE

Funkcja "wyswietl_samochody_w_cenie" przyjmuje dwa parametry - p_cena_od i p_cena_do, które określają zakres cenowy w jakim mają być wyszukane samochody. Następnie kursor c_samochody pobiera z bazy danych informacje o samochodach spełniających warunek z zakresu cenowego. W dalszej kolejności funkcja sprawdza czy podane parametry znajdują się w odpowiednim zakresie, jeśli nie rzuca wyjątkiem. Następnie iteruje przez kursor i buduje string z danymi o samochodach i zwraca go jako wynik. Jeśli w bazie danych nie ma pasujących samochodów rzuci wyjątkiem.

```oracle_plsql
create or replace FUNCTION wyswietl_samochody_w_cenie
(p_cena_od NUMBER, p_cena_do NUMBER) RETURN VARCHAR2 AS
v_result VARCHAR2(4000);
CURSOR c_samochody IS
   SELECT s.id_pojazdu, s.rocznik, s.cena, m.nazwa marka,
    mo.nazwa_modelu model, kp.kraj, n.nadwozie, p.rodzaj,
    ko.nazwa
    FROM samochody s
    INNER JOIN modele mo ON s.id_modelu = mo.id_modelu
    INNER JOIN marki m ON mo.id_marki = m.id_marki
    INNER JOIN krajpochodzenia kp ON s.id_kraju = kp.id_kraju
    INNER JOIN kolory ko ON s.id_koloru = ko.id_koloru
    INNER JOIN paliwo p ON s.id_paliwa = p.id_paliwa
    INNER JOIN nadwozie n ON s.id_nadwozie = n.id_nadwozie
    WHERE s.cena BETWEEN p_cena_od AND p_cena_do;
BEGIN
    IF p_cena_od < 1000 OR p_cena_do > 500000 THEN
    RAISE VALUE_ERROR;
    ELSE
        v_result := ('Lista samochodów w zakresie cenowym ' ||
         p_cena_od || 'zł a ' || p_cena_do || 'zł:' || '<br>');
        FOR r_samochody IN c_samochody LOOP
            v_result := (v_result || r_samochody.marka || ' ' ||
             r_samochody.model || ', rocznik ' || r_samochody.rocznik ||
              ', cena ' || r_samochody.cena || ' zł' || ', kraj ' ||
               r_samochody.kraj || ', nadwozie ' || r_samochody.nadwozie ||
            ', paliwo ' || r_samochody.rodzaj || ', kolor ' ||
             r_samochody.nazwa || '. ' || '<br>');
        END LOOP;
    RETURN v_result;
    END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
raise_application_error(-20002, 'Brak samochodów w bazie danych');
WHEN VALUE_ERROR THEN
raise_application_error(-20005, 'Podane liczby muszą być w zakresie 1000 - 500 000');
END;
```

12. WYSZUKAJ_SAMOCHODY

Funkcja "wyszukaj_samochody" przyjmuje pięć parametrów: marka, model, rocznik od, rocznik do i cena od, cena do. Funkcja wykonuje kwerendę w bazie danych, która zwraca informacje o samochodach spełniających podane kryteria. Jeśli marka lub model nie zostanie znaleziony w bazie danych, funkcja zgłasza błąd. Jeśli podane lata są błędne lub cena poza zakresem, funkcja również zgłasza błąd. Jeśli zostanie znaleziony co najmniej jeden samochód spełniający kryteria, funkcja zwraca listę tych samochodów.

```oracle_plsql
create or replace FUNCTION wyszukaj_samochody(p_marka VARCHAR2,
p_model VARCHAR2, p_rocznik_od NUMBER, p_rocznik_do NUMBER,
p_cena_od NUMBER, p_cena_do NUMBER) RETURN VARCHAR2
AS
v_result VARCHAR2(4000);
v_count NUMBER;
v_count1 NUMBER;
brak exception;
brak1 exception;
brak2 exception;
brak3 exception;
CURSOR c_samochody IS
   SELECT s.id_pojazdu, s.rocznik, s.cena, m.nazwa marka, mo.nazwa_modelu model,
          kp.kraj, n.nadwozie, p.rodzaj, ko.nazwa
    FROM samochody s
    INNER JOIN modele mo ON s.id_modelu = mo.id_modelu
    INNER JOIN marki m ON mo.id_marki = m.id_marki
    INNER JOIN krajpochodzenia kp ON s.id_kraju = kp.id_kraju
    INNER JOIN kolory ko ON s.id_koloru = ko.id_koloru
    INNER JOIN paliwo p ON s.id_paliwa = p.id_paliwa
    INNER JOIN nadwozie n ON s.id_nadwozie = n.id_nadwozie
    WHERE m.nazwa LIKE '%' || p_marka || '%'
      AND mo.nazwa_modelu LIKE '%' || p_model || '%'
      AND s.rocznik BETWEEN p_rocznik_od AND p_rocznik_do
      AND s.cena BETWEEN p_cena_od AND p_cena_do;
BEGIN
    SELECT COUNT(*) INTO v_count
    FROM marki
    WHERE nazwa = p_marka;

    SELECT COUNT(*) INTO v_count1
    FROM modele
    WHERE nazwa_modelu = p_model;

    IF v_count = 0 THEN
       RAISE brak;
    ELSIF v_count1 = 0 THEN
        RAISE brak1;
            ELSIF p_rocznik_od < 1900 OR p_rocznik_do > 2024 THEN
                RAISE brak2;
                ELSIF p_cena_od < 1000 OR p_cena_do > 500000 THEN
                        RAISE brak3;
                    ELSE
```

```
                    v_result := ('Samochody spelniajace kryteria:' || '<br>');
                    FOR r_samochody IN c_samochody LOOP
                        v_result := (v_result || r_samochody.marka || ' ' ||
                         r_samochody.model || ', rocznik ' || r_samochody.rocznik ||
                          ', cena ' || r_samochody.cena || ' zł' || ', kraj ' ||
                           r_samochody.kraj || ', nadwozie ' || r_samochody.nadwozie ||
                        ', paliwo ' || r_samochody.rodzaj || ', kolor ' ||
                         r_samochody.nazwa || '. ' || '<br>');
                    END LOOP;
                RETURN v_result;
    END IF;

EXCEPTION
WHEN NO_DATA_FOUND THEN
RETURN 'Nie znaleziono samochodów spełniających podane kryteria.';
WHEN brak THEN
raise_application_error(-20005, 'Nie znaleziono marki:' || p_marka || ' w naszym komisie.');
WHEN brak1 THEN
raise_application_error(-20004, 'Nie znaleziono modelu:' || p_model || ' w naszym komisie.');
WHEN brak2 THEN
raise_application_error(-20004, 'Podane lata są błędne.');
WHEN brak3 THEN
raise_application_error(-20004, 'Cena musi byc w zakresie 1000 - 500 000.');
END;
```

5. Pakiety

Zostały utworzone 3 pakiety, w którego skład wchodzą:

1. PAKIET1

Procedury: dodaj_klienta; dodaj_samochod; edytuj_klienta; średnia_cena_ostatniego

Funkcje:cena_z_rabatem, czy_klient_kupil_samochod, czy_samochod_dostepny, najczesniej_kupowana_marka

2. PAKIET2

Procedury: szczegoly_transakcji usun_samochod; wyświetl_klientow; wyświetl_samochody_przebiegu

Funkcje: najczestsze_kolory_samochodow; procentowy_udzial_marki; srednia_cena_marki; sredni_rocznik_samochodow_danej_marki

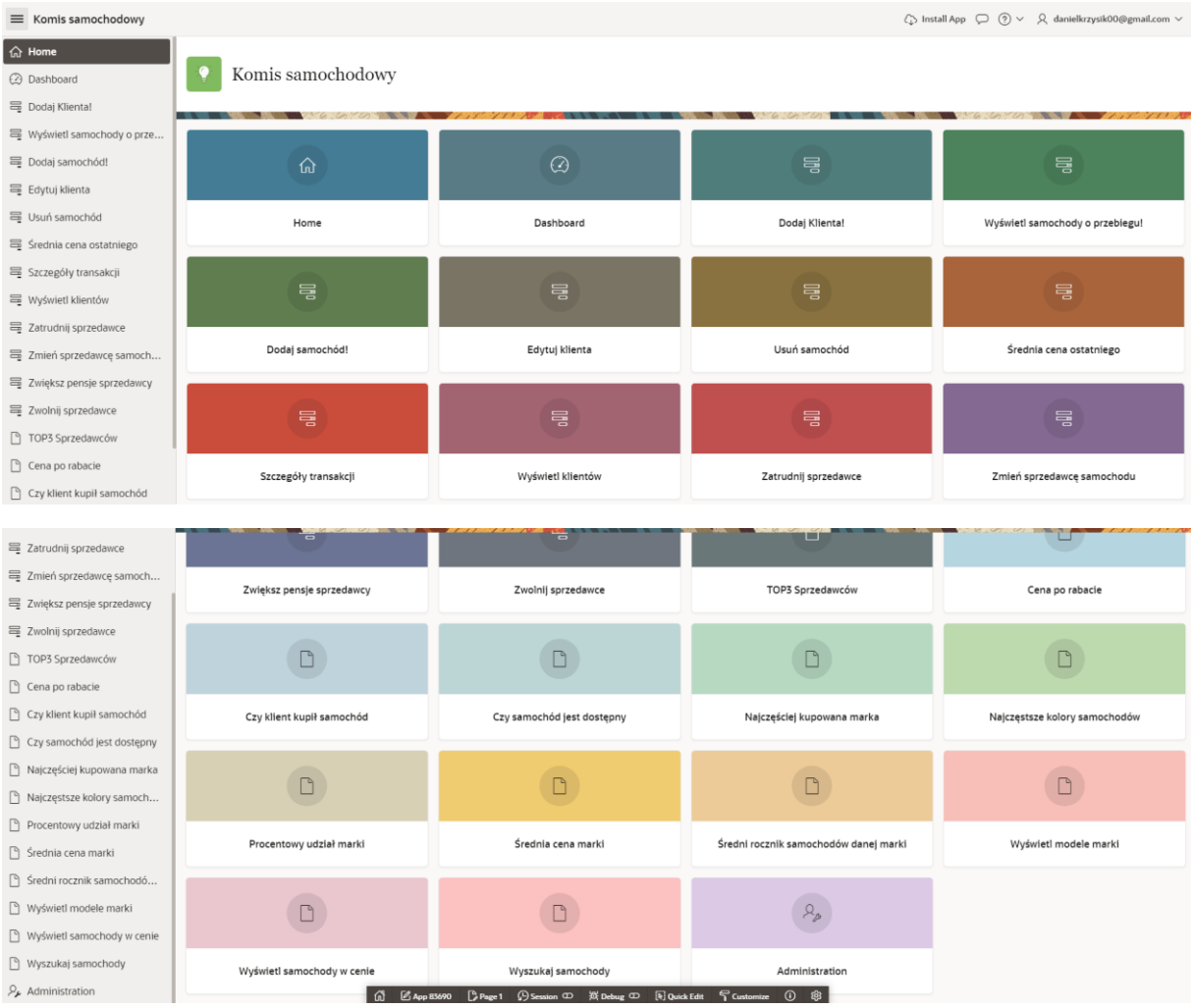
3. PAKIET3

Procedury: zatrudnij_sprzedawce; zmien_sprzedawce_samochodu; zwiekszpensjesprzedawcy; zwolnij_sprzedawce

Funkcje: top_3_najlepszych_sprzedawcow; wyświetl_modele_marki; wyświetl_samochody_w_cenie; wyszukaj_samochody

6. Aplikacja

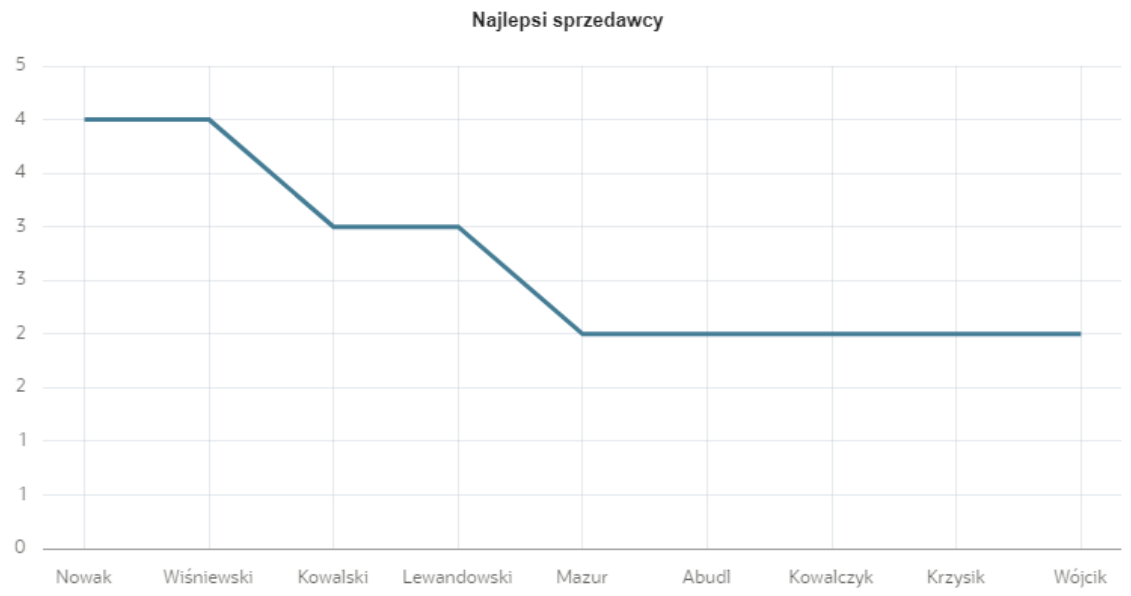
1. HOME:



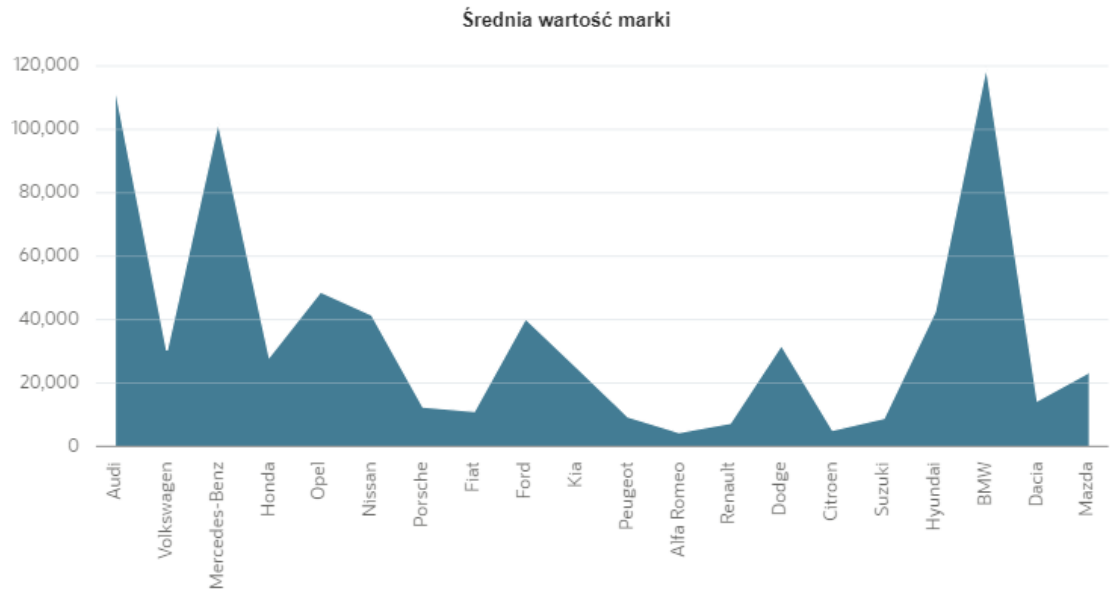
2. DASHBOARD

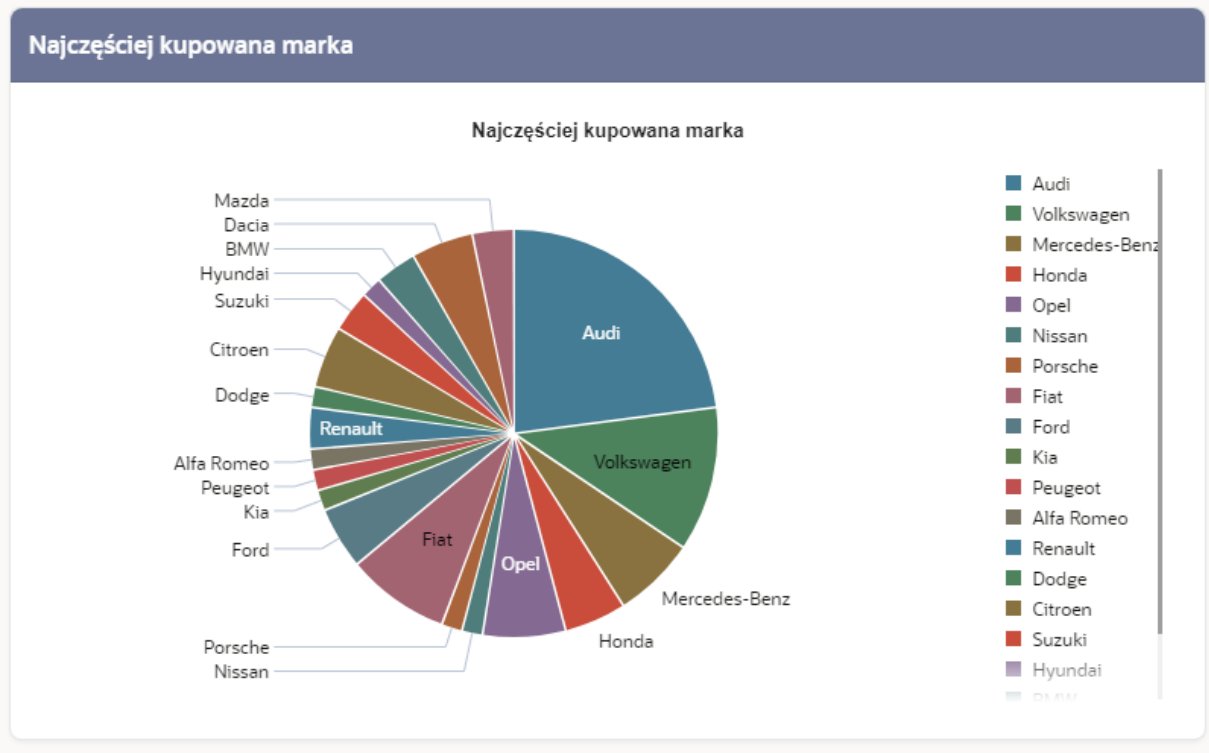
Wygląd strony:

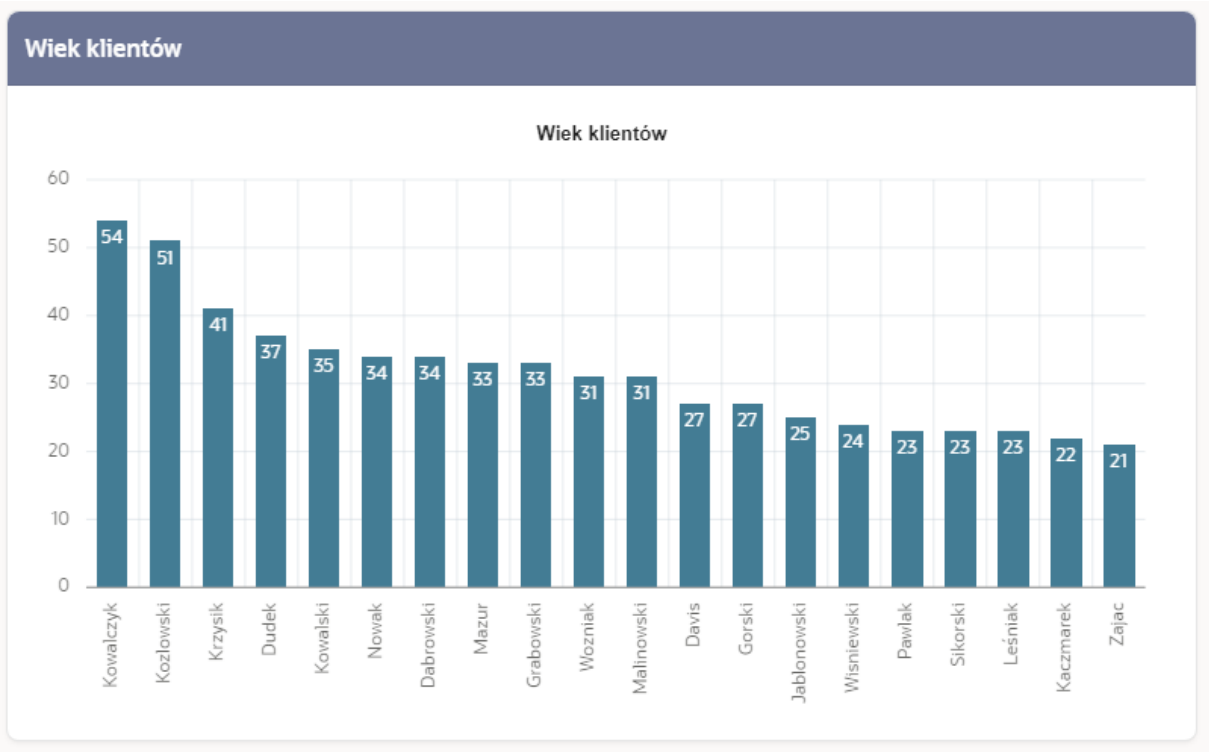
Najlepsi sprzedawcy



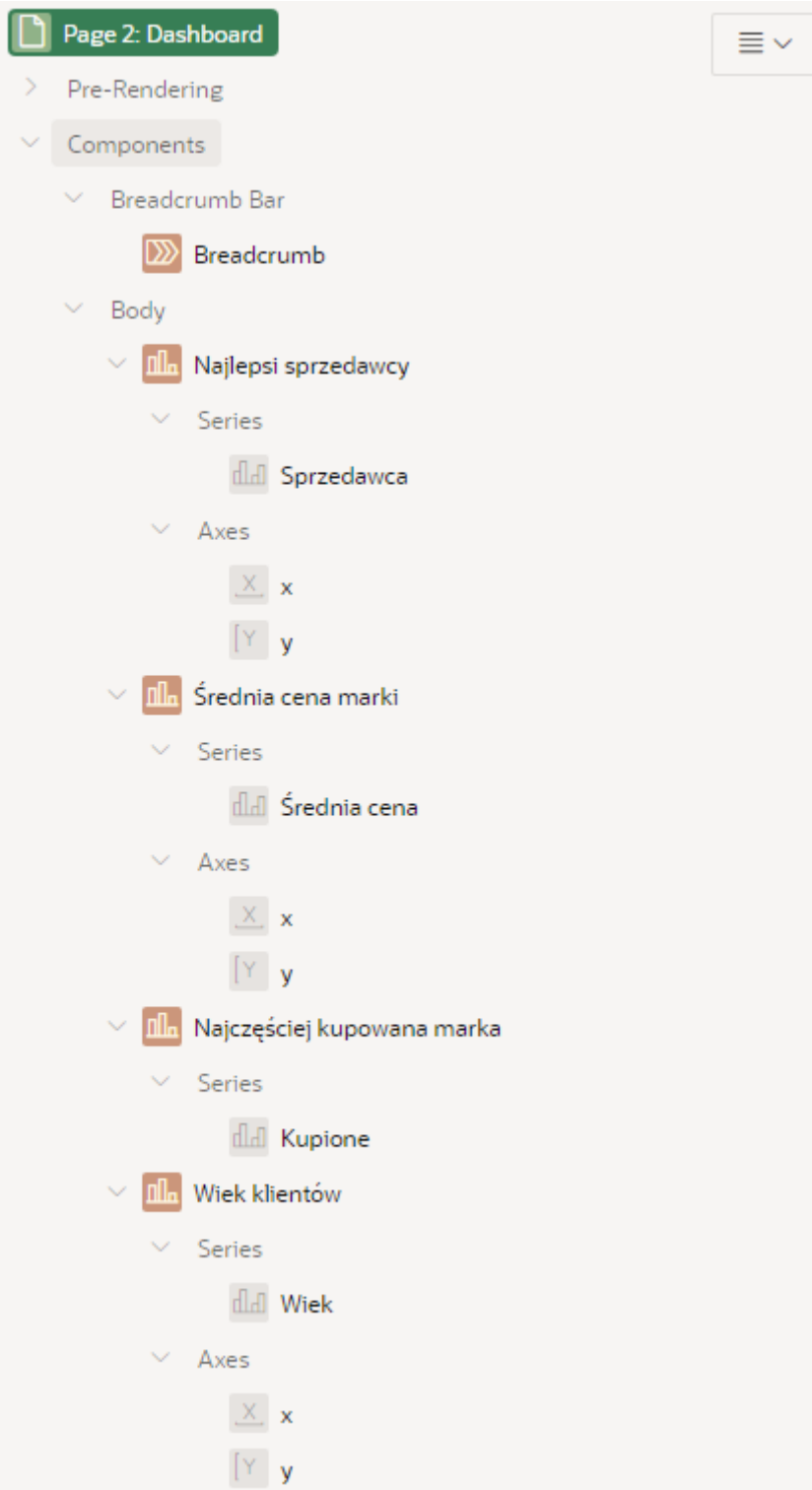
Średnia cena marki







Page Designer:



Kod do pierwszego wykresu:

```
SELECT s.nazwisko, s.imie, COUNT(t.id_transakcji) as ilosc_samochodow
       FROM transakcje t
       INNER JOIN sprzedawcy s ON t.id_sprzedawcy = s.id_sprzedawcy
       GROUP BY s.nazwisko, s.imie
       ORDER BY ilosc_samochodow DESC;
```

Kod do drugiego wykresu:

```
SELECT round(AVG(s.cena),0) as Srednia_Wartosc, m.nazwa
FROM samochody s INNER JOIN modele mo ON
s.id_modelu = mo.id_modelu
INNER JOIN marki m ON mo.id_marki = m.id_marki
GROUP BY m.nazwa
```

Kod do trzeciego wykresu:

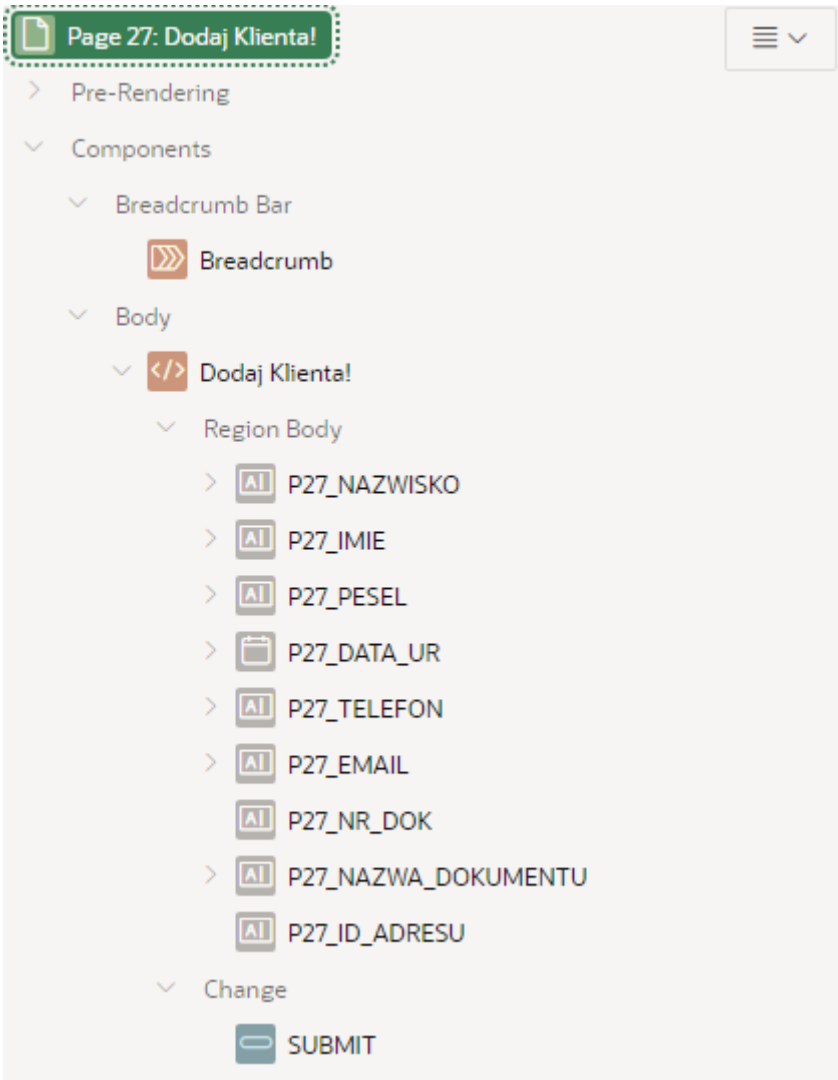
```
SELECT ma.nazwa marka_samochodu, COUNT(*) ilosc
    FROM samochody s INNER JOIN
    modele m ON s.id_modelu = m.id_modelu
    INNER JOIN marki ma ON m.id_marki = ma.id_marki
    GROUP BY ma.nazwa;
```

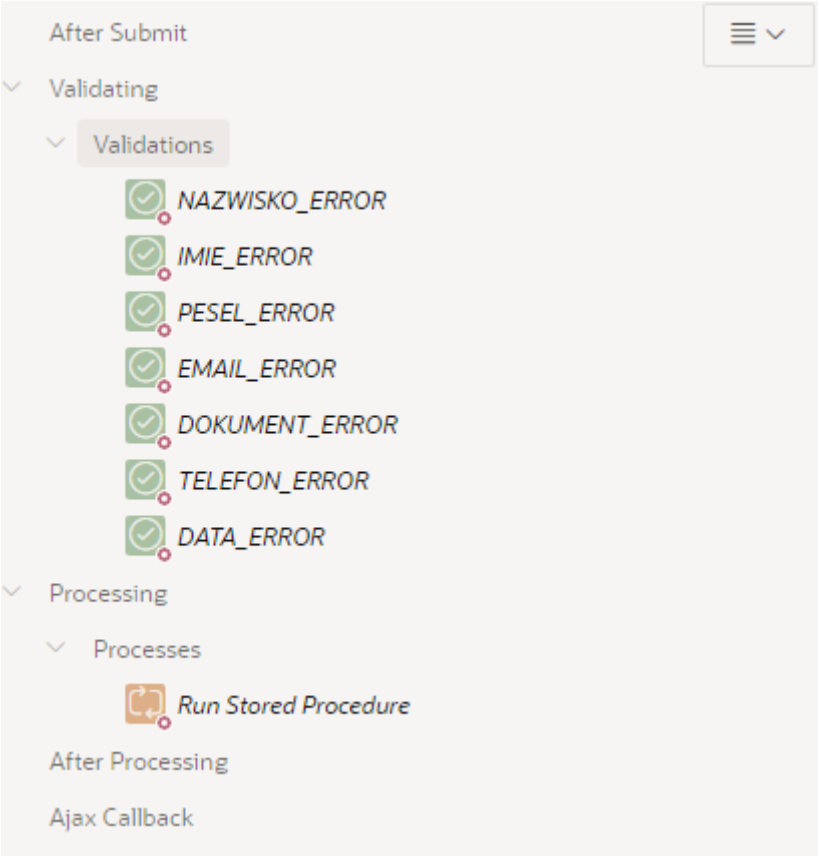
Kod do czwartego wykresu:

```
SELECT nazwisko as Nazwisko, imie as Imie,
TRUNC(TO_NUMBER(SYSDATE - TO_DATE(data_ur))
 / 365.25) AS Wiek FROM klienci
ORDER BY Wiek DESC;
```

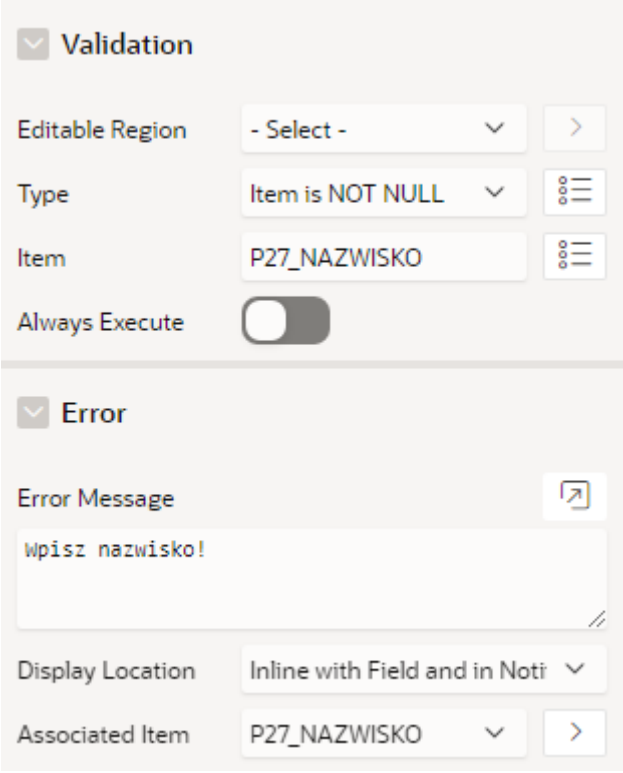
3. Dodaj klienta

Page Designer:


Page 27: Dodaj Klienta!
Pre-Rendering
Components
Breadcrumb Bar
Breadcrumb
Body
Dodaj Klienta!
Region Body
P27_NAZWISKO
P27_IMIE
P27_PESEL
P27_DATA_UR
P27_TELEFON
P27_EMAIL
P27_NR_DOK
P27_NAZWA_DOKUMENTU
P27_ID_ADRESU
Change
SUBMIT



Validations:



Validation

Editable Region: - Select -

Type: Item is NOT NULL

Item: P27_IMIE

Always Execute: [ ]

Error

Error Message

```
Wpisz imie!
```

Display Location: Inline with Field and in Noti

Associated Item: P27_IMIE

Validation

Editable Region: - Select -

Type: Function Body (retu

Language: PL/SQL

PL/SQL Function Body Returning Error Text

```
begin
IF LENGTH(:P27_PESEL) <> 11 THEN
    return 'Błędny pesel!';
ELSE
    null;
END IF;
end;
```

Validation

Editable Region: - Select -

Type: Function Body (retu

Language: PL/SQL

PL/SQL Function Body Returning Error Text

```
begin
IF INSTR(:P27_EMAIL, '@gmail.com') < 1 THEN
    return 'Niepoprawny adres e-mail. Musi
zawierać końcówkę @gmail.com';
ELSE
    null;
END IF;
end;
```

Validation

Editable Region: - Select -

Type: Function Body (retu

Language: PL/SQL

PL/SQL Function Body Returning Error Text

```
begin
IF :P27_NAZWA_DOKUMENTU NOT IN ('Paszport',
'Dowod osobisty') THEN
return 'Wybierz poprawny dokument!';
ELSE
    null;
END IF;
end;
```

Validation

Editable Region - Select -

Type Function Body (retu

Language PL/SQL

PL/SQL Function Body Returning Error Text

```
begin
IF NOT REGEXP_LIKE(:P27_TELEFON, '^[0-9]+$')
THEN
return 'Numer telefonu może składać się tylko
z cyfr.';
ELSIF LENGTH(:P27_TELEFON) <> 9 THEN
return 'Numer musi sie skladac z 9 znakow';
ELSE
    null;
END IF;
end;
```

Validation

Editable Region - Select -

Type Function Body (retu

Language PL/SQL

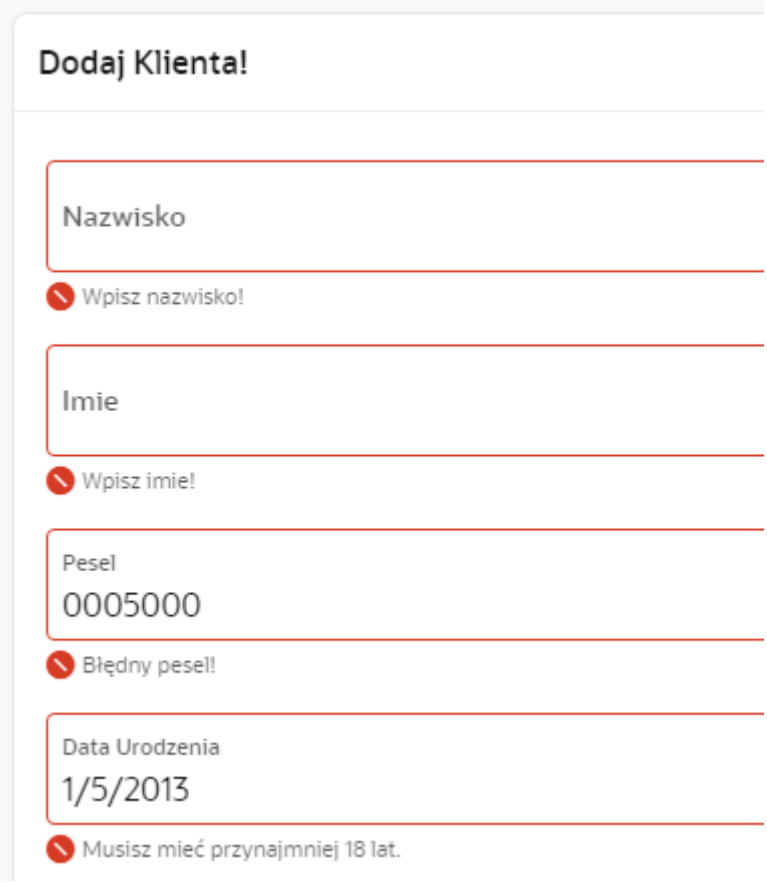
PL/SQL Function Body Returning Error Text

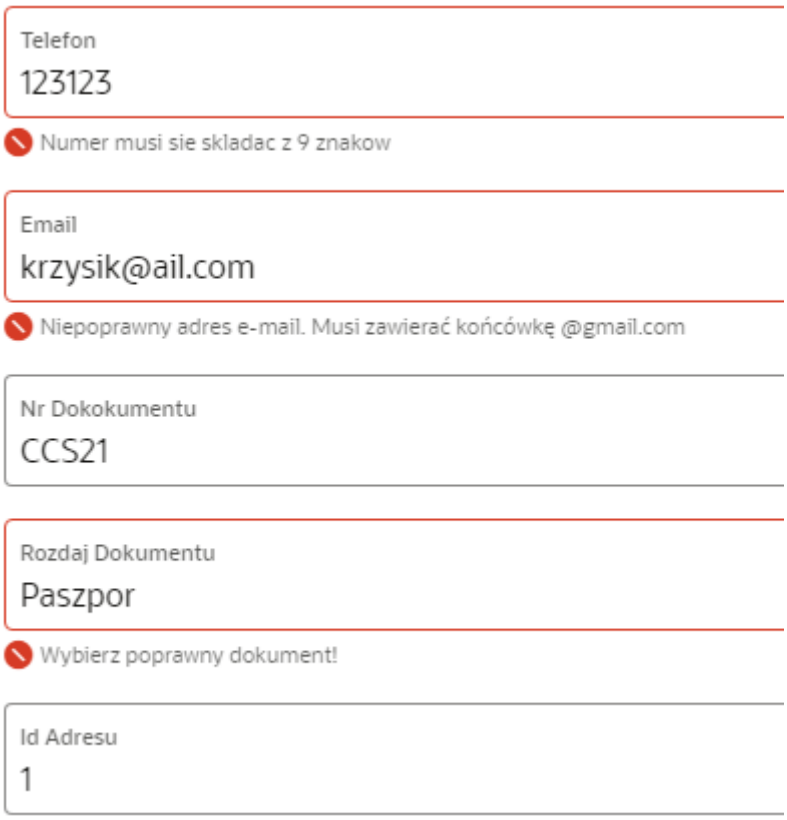
```
begin
IF (trunc(months_between(sysdate,
to_date(:P27_DATA_UR,'mm/dd/yyyy'))/12) < 18 )
THEN
return 'Musisz mieć przynajmniej 18 lat.';
ELSE
null;
END IF;
end;
```

Kod:

```
pakiet1.dodaj_klienta(
    :P27_NAZWISKO,
    :P27_IMIE,
    :P27_PESEL,
    :P27_DATA_UR,
    :P27_TELEFON,
    :P27_EMAIL,
    :P27_NR_DOK,
    :P27_NAZWA_DOKUMENTU,
    :P27_ID_ADRESU,
    :P27_NEW
);
```

Dodanie klienta:





Poprawne wpisanie danych:

Dodaj Klienta!

Nazwisko
Krzysik

Imie
Daniel

Pesel
00005000000

Data Urodzenia
1/5/2000

Telefon
123123123

Email
krzysik@gmail.com

Nr Dokokumentu
CCS21

Rozdaj Dokumentu
Paszport

Id Adresu
1

Install App

Pomyślnie dodano klienta!

Sprawdzenie czy klient został dodany do tabeli:

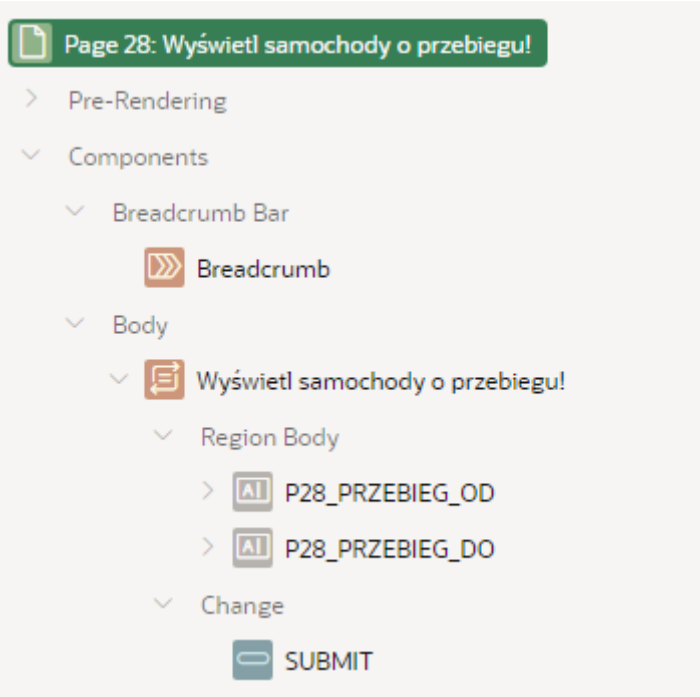
```
1   select * from klienci ORDER BY ID_KLIENTA DESC;
```

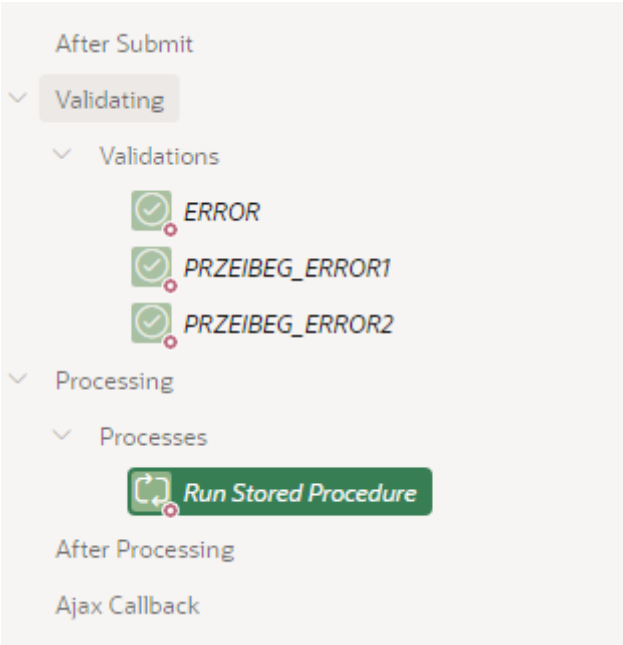
Results | Explain | Describe | Saved SQL | History

ID_KLIENTA	NAZWISKO	IMIE	PESEL	DATA_UR
24	Krzysik	Daniel	00005000000	01/05/2000

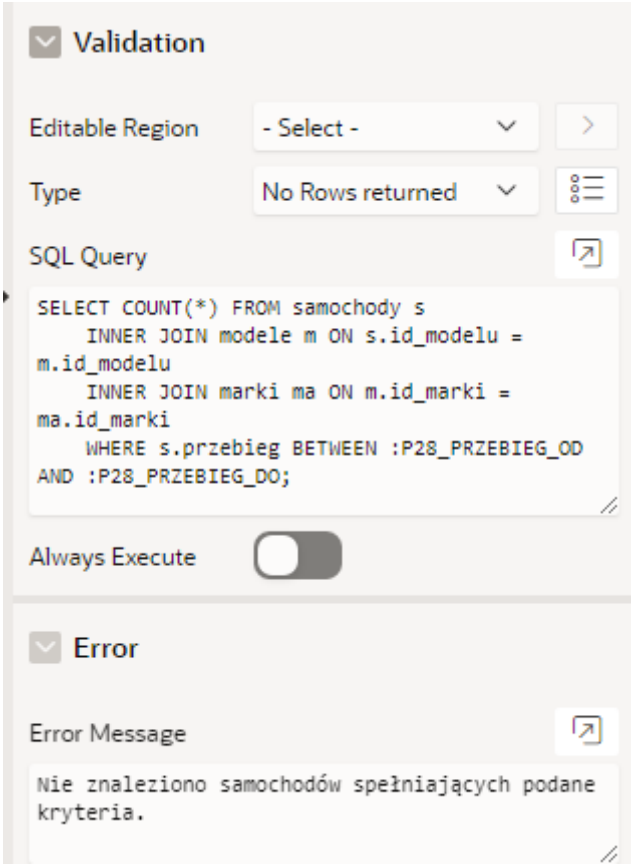
4. Wyświetl samochody o przebiegu

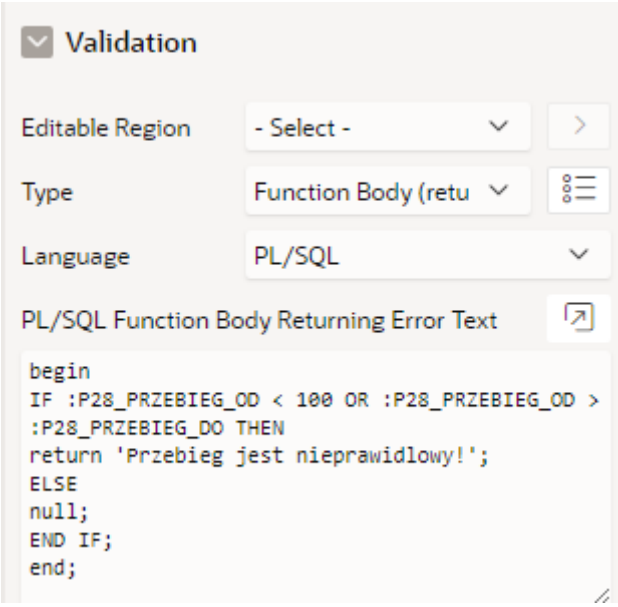
Page Designer:

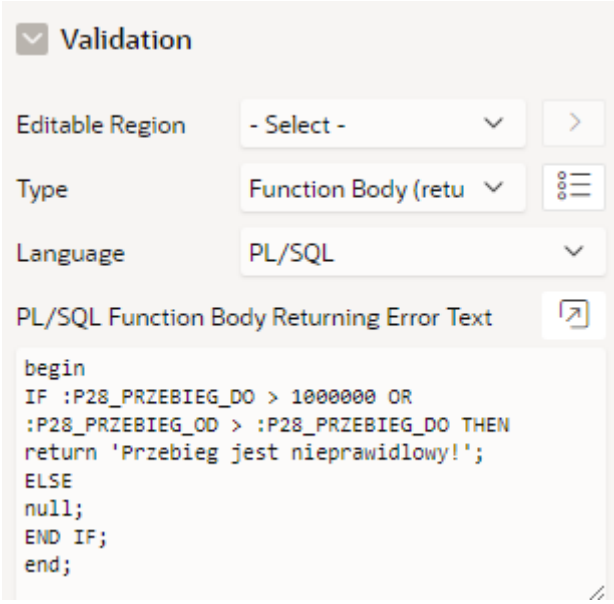




Validations:





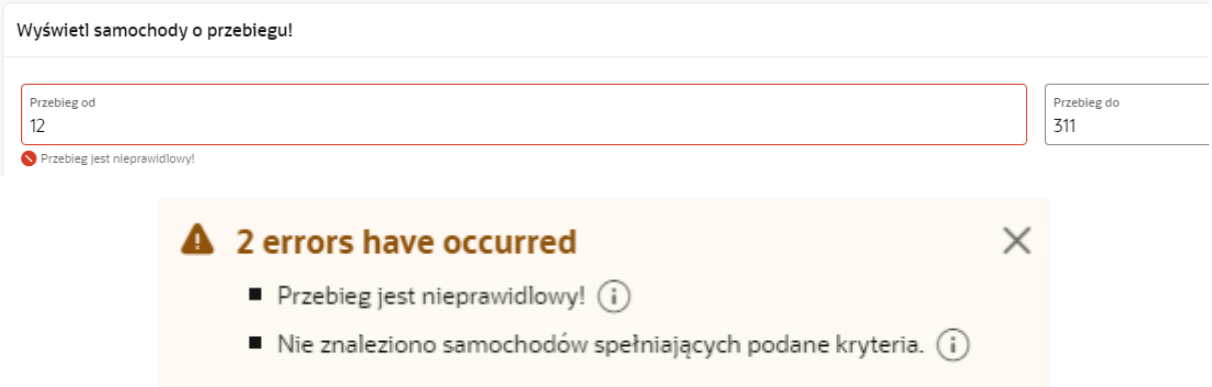


Kod:

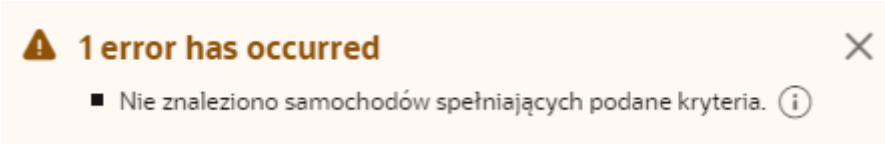
```
PAKIET2.wyswietl_samochody_o_przebiegu(
    :P28_PRZEBIEG_OD,
    :P28_PRZEBIEG_DO
);
```

Wygląd:

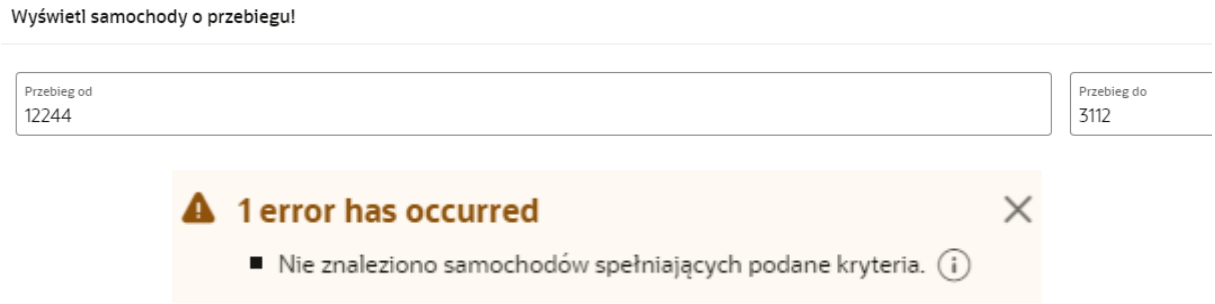
Pierwszy przypadek



Drugi przypadek



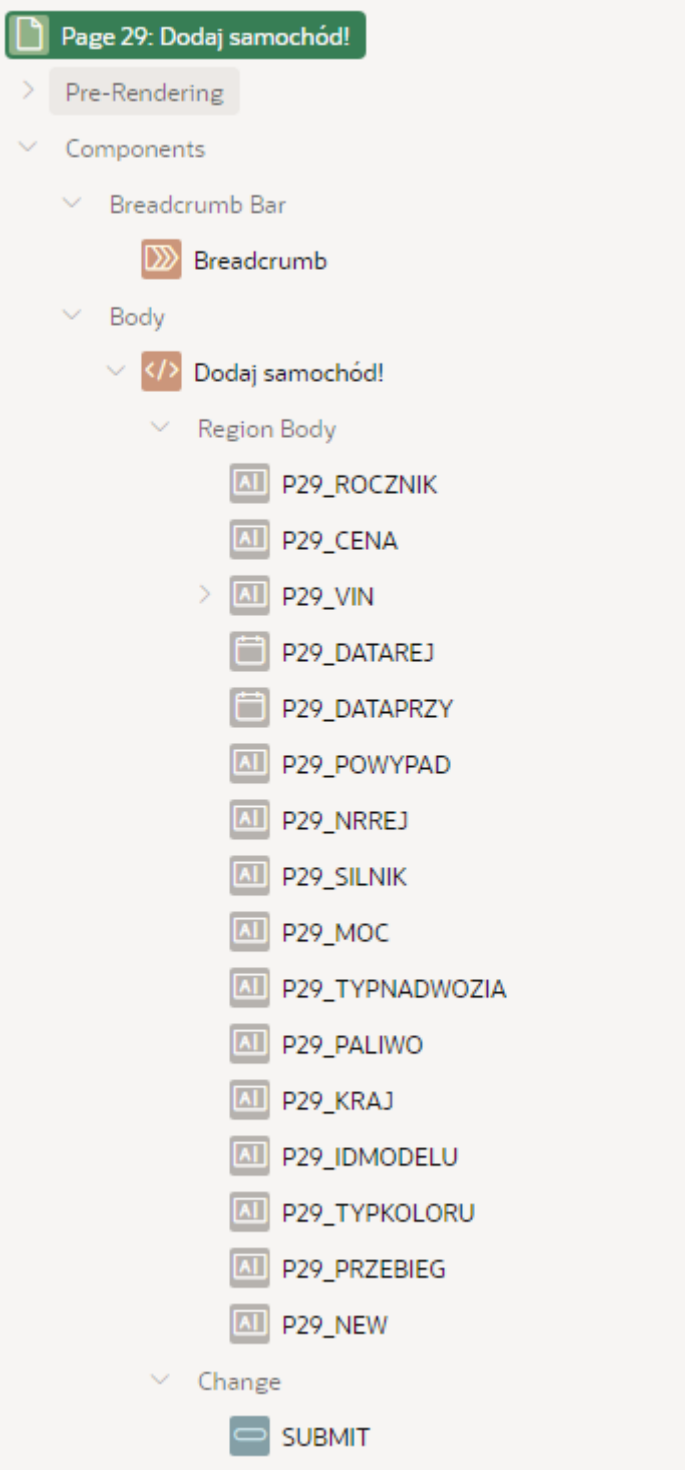
Trzeci przypadek

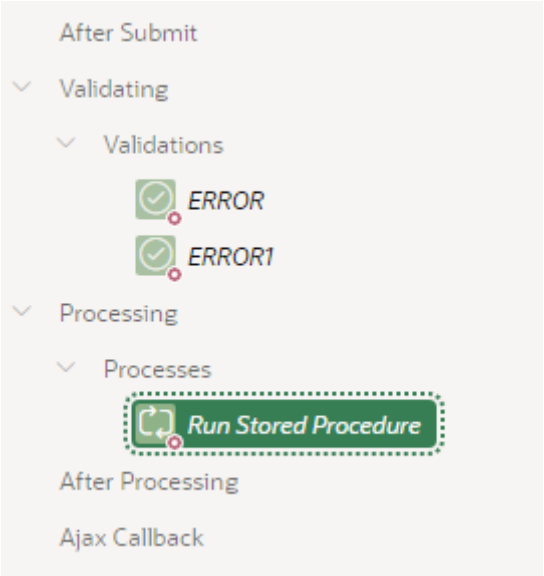


Gdy prawidłowo wpiszemy

5. Dodaj samochód

Page Designer:

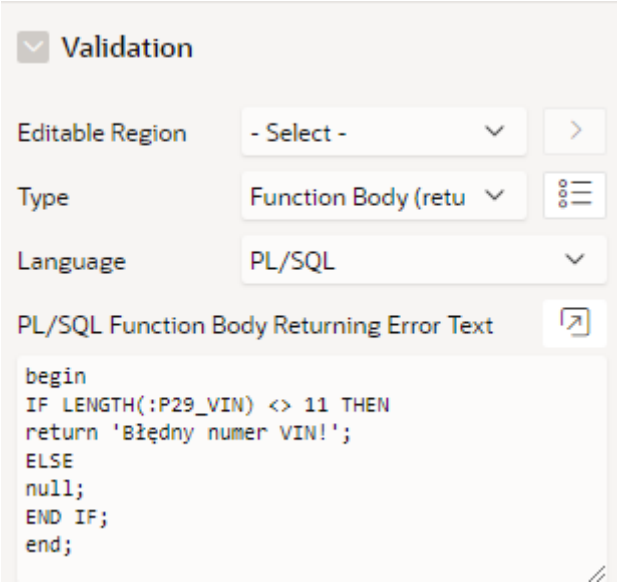

Page 29: Dodaj samochód!
Pre-Rendering
Components
Breadcrumb Bar
Breadcrumb
Body
Dodaj samochód!
Region Body
P29_ROCZNIK
P29_CENA
P29_VIN
P29_DATAREJ
P29_DATAPRZY
P29_POWYPAD
P29_NRREJ
P29_SILNIK
P29_MOC
P29_TYPNADWOZIA
P29_PALIWO
P29_KRAJ
P29_IDMODELU
P29_TYPKOLORU
P29_PRZEBIEG
P29_NEW
Change
SUBMIT

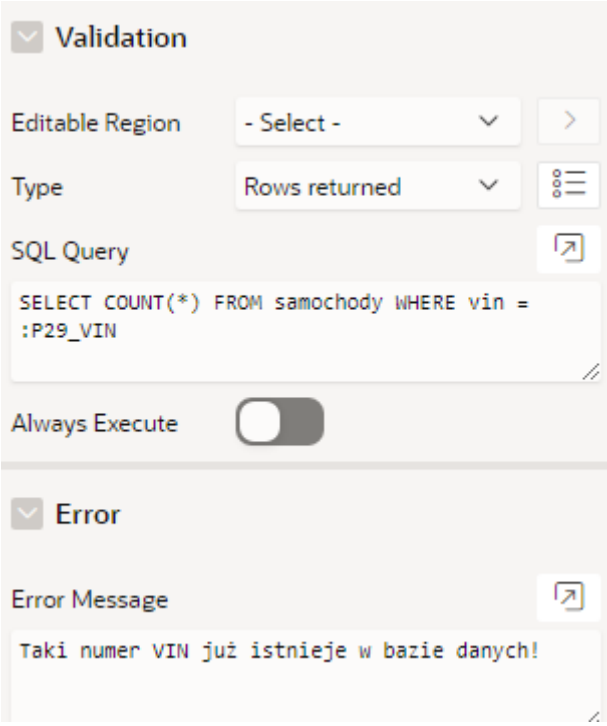


Kod:

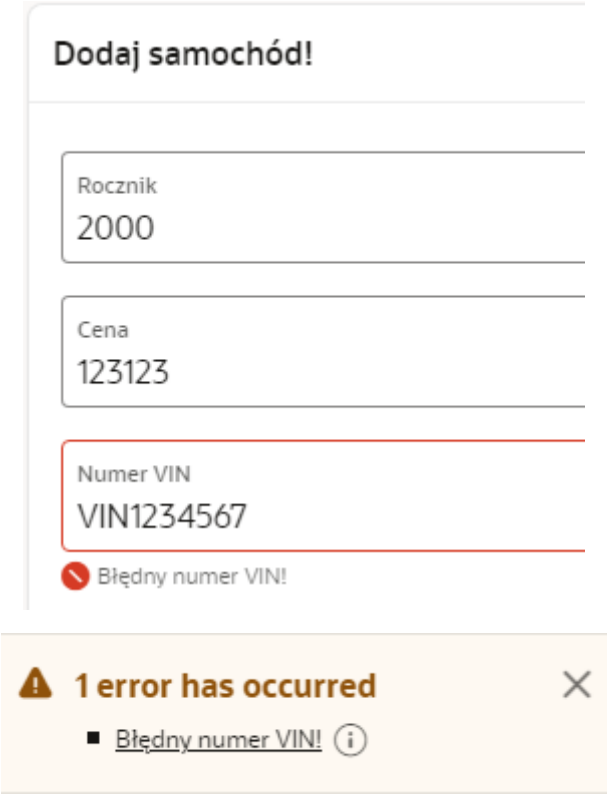
```
pakiet1.dodaj_samochod(
    :P29_ROCZNIK,
    :P29_CENA,
    :P29_VIN,
    :P29_DATAREJ,
    :P29_DATAPRZY,
    :P29_POWYPAD,
    :P29_NRREJ,
    :P29_SILNIK,
    :P29_MOC,
    :P29_TYPNADWOZIA,
    :P29_PALIWO,
    :P29_KRAJ,
    :P29_IDMODELU,
    :P29_TYPKOLORU,
    :P29_PRZEBIEG,
    :P29_NEW
);
```

Validations:

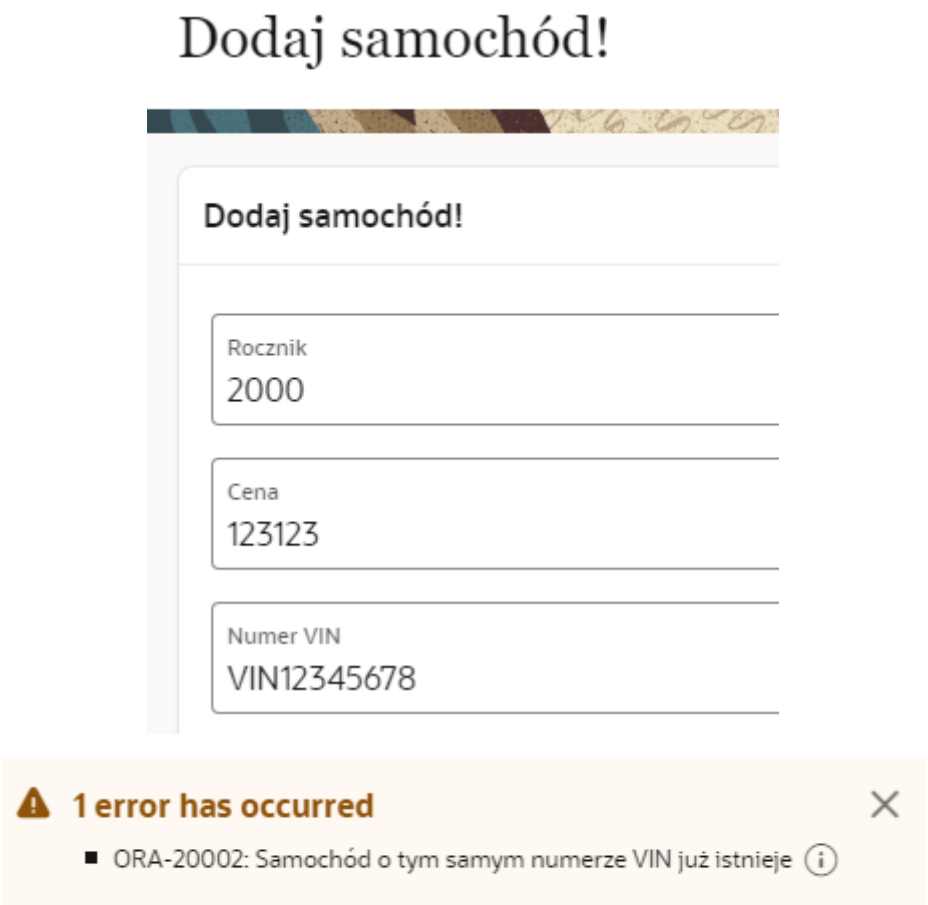




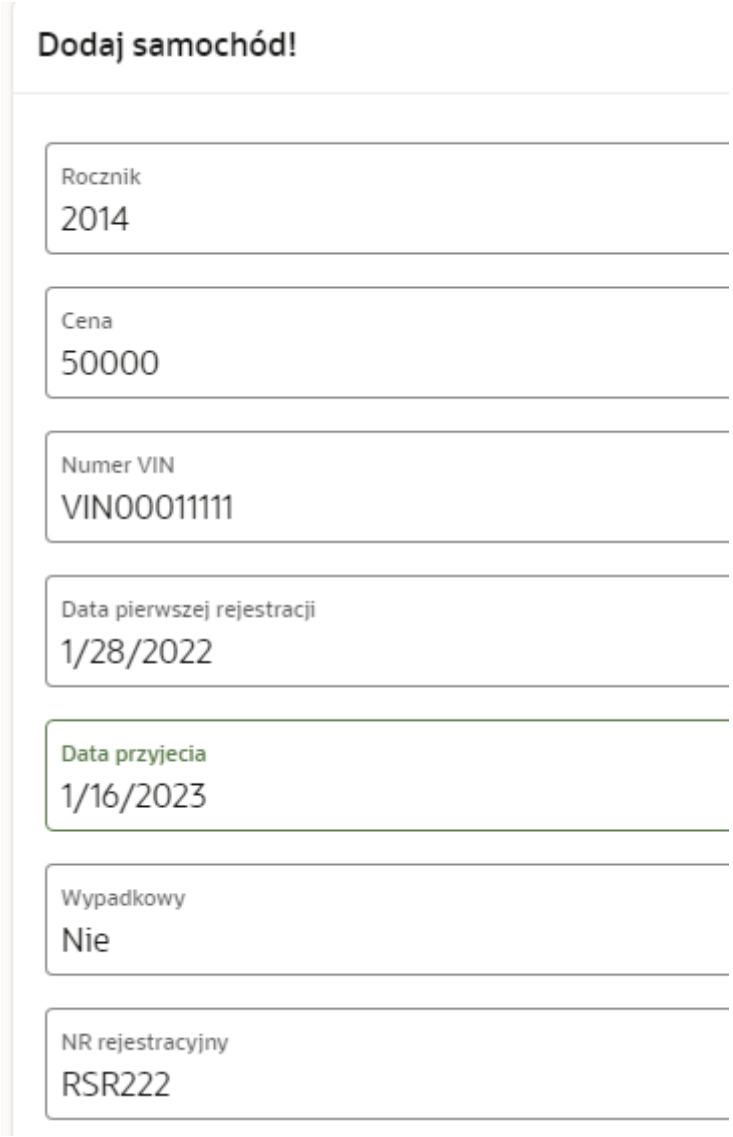
Wygląd strony:



Drugi przypadek



Poprawne wpisane dane:



Silnik
2.0

Moc
220

IDnadwozia
2

IDpaliwa
1

IDkraju
3

IDmodelu
4

IDkoloru
2

Przebieg
134000

Pomyślnie dodano samochód!

Jak widać poprawnie został dodany samochód do naszej bazy.

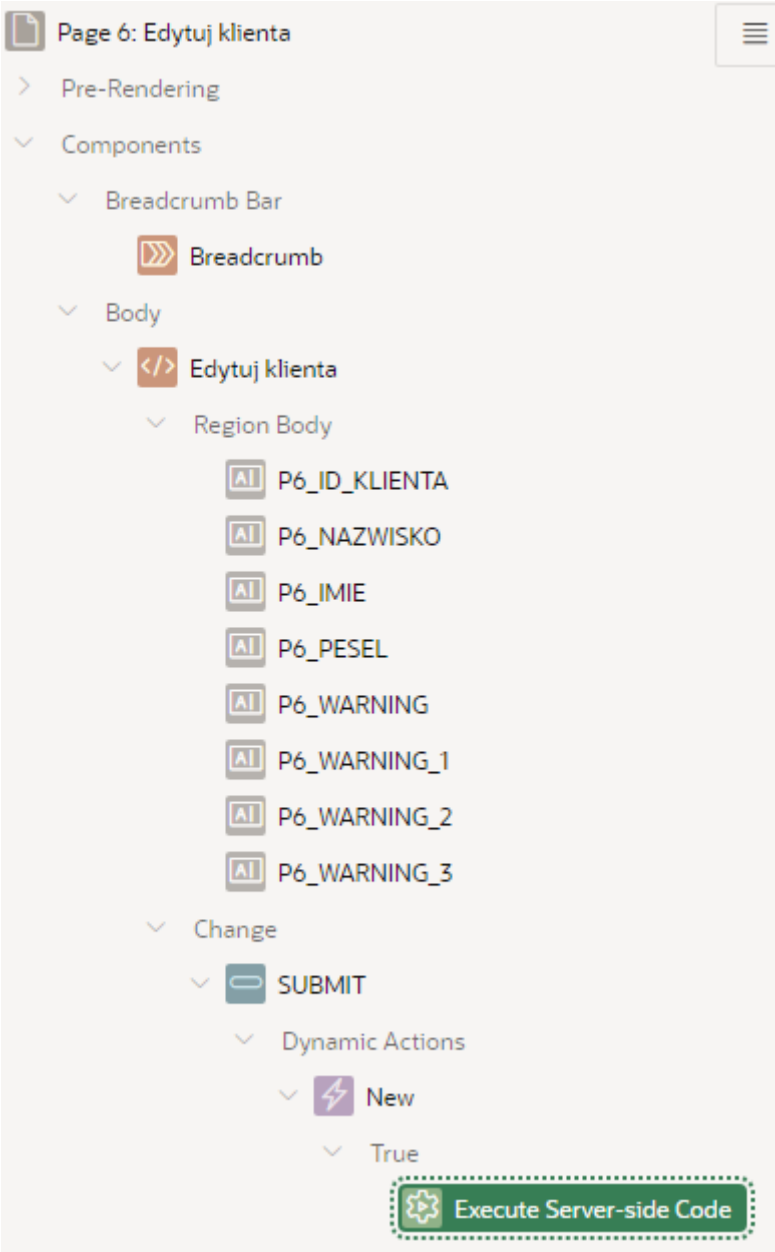
```
1   select * from samochody where vin like 'VIN00011111';
```

Results | Explain | Describe | Saved SQL | History

ID_POJAZDU	ROCZNIK	CENA	VIN	DATA_REJ
62	2014	50000	VIN00011111	01/28/2022

6. Edytuj klienta

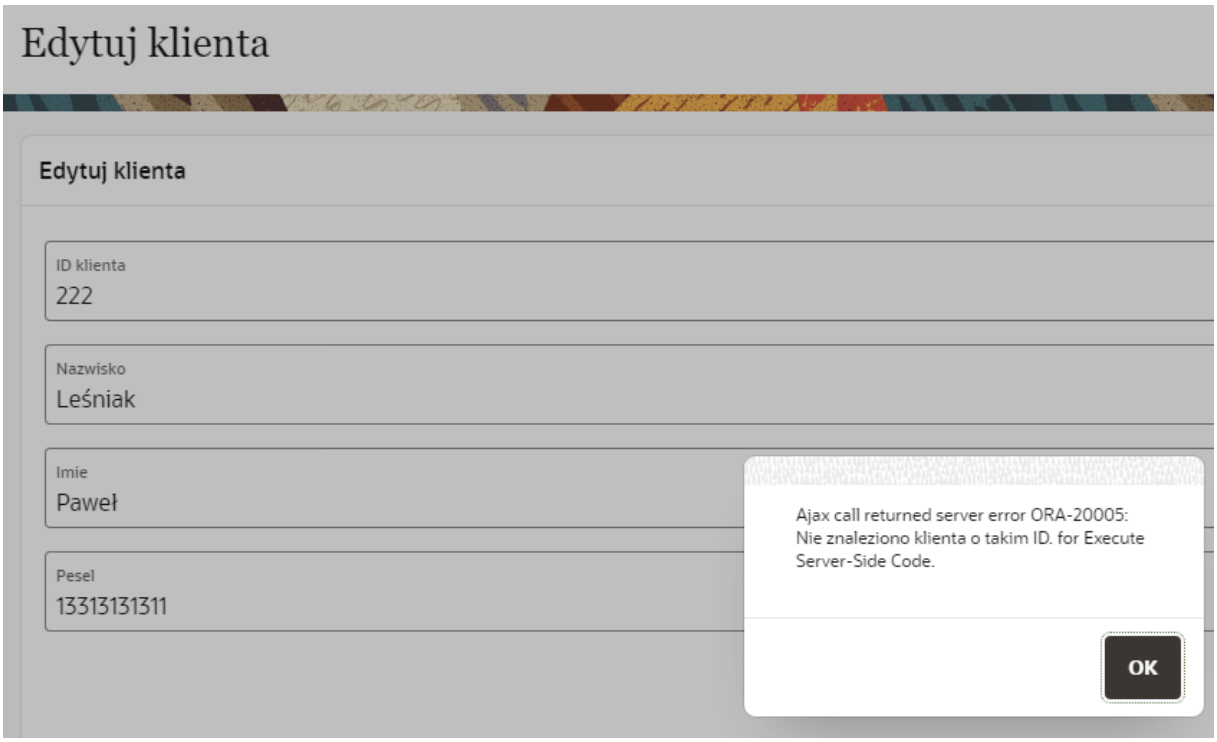
Page Designer:

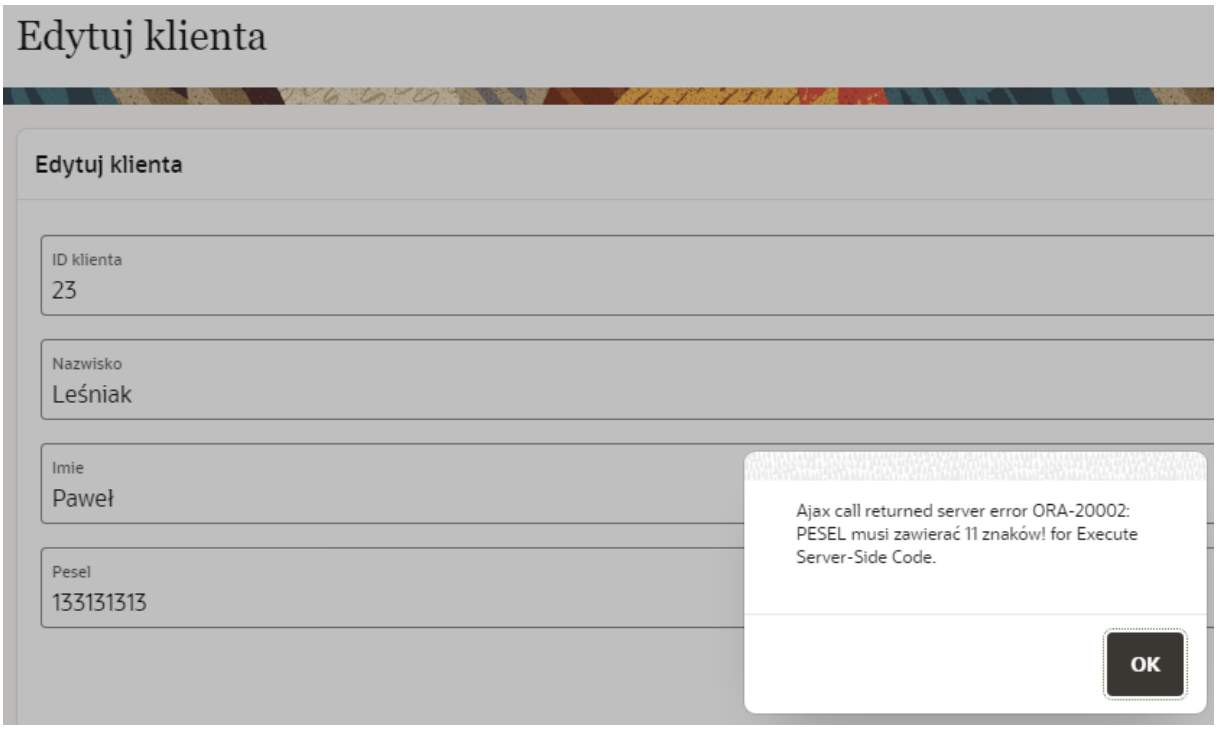

Page 6: Edytuj klienta
Pre-Rendering
Components
Breadcrumb Bar
Breadcrumb
Body
Edytuj klienta
Region Body
P6_ID_KLIENTA
P6_NAZWISKO
P6_IMIE
P6_PESEL
P6_WARNING
P6_WARNING_1
P6_WARNING_2
P6_WARNING_3
Change
SUBMIT
Dynamic Actions
New
True
Execute Server-side Code

Kod:

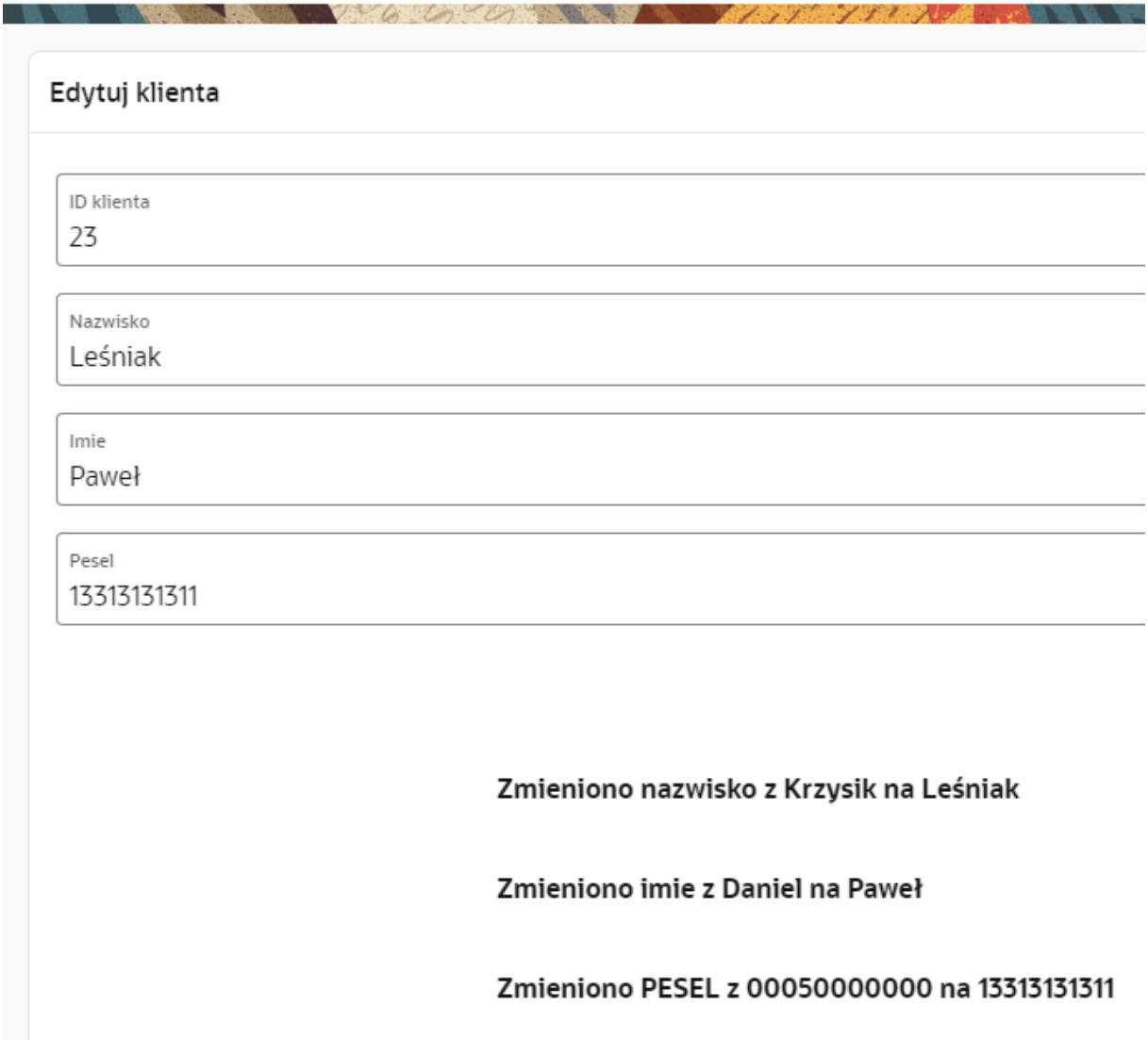
```
pakiet1.edytuj_klienta(
    :P6_ID_KLIENTA,
    :P6_NAZWISKO,
    :P6_IMIE,
    :P6_PESEL,
    :P6_WARNING,
    :P6_WARNING_1,
    :P6_WARNING_2,
    :P6_WARNING_3
);
```

Gdy wpiszemy niepoprawne ID:




Edytuj klienta
Edytuj klienta
ID klienta
23
Nazwisko
Leśniak
Imie
Paweł
Pesel
133131313
Ajax call returned server error ORA-20002: PESEL musi zawierać 11 znaków! for Execute Server-Side Code.
OK

Edytuj klienta



Nie musimy zmieniać wszystkich wartości można tylko jedną:

Edytuj klienta

ID klienta
23

Nazwisko

Imie

Pesel
00001100119

Zmieniono PESEL z 13313131311 na 00001100119

Jak widać dane klienta zostały zmienione:

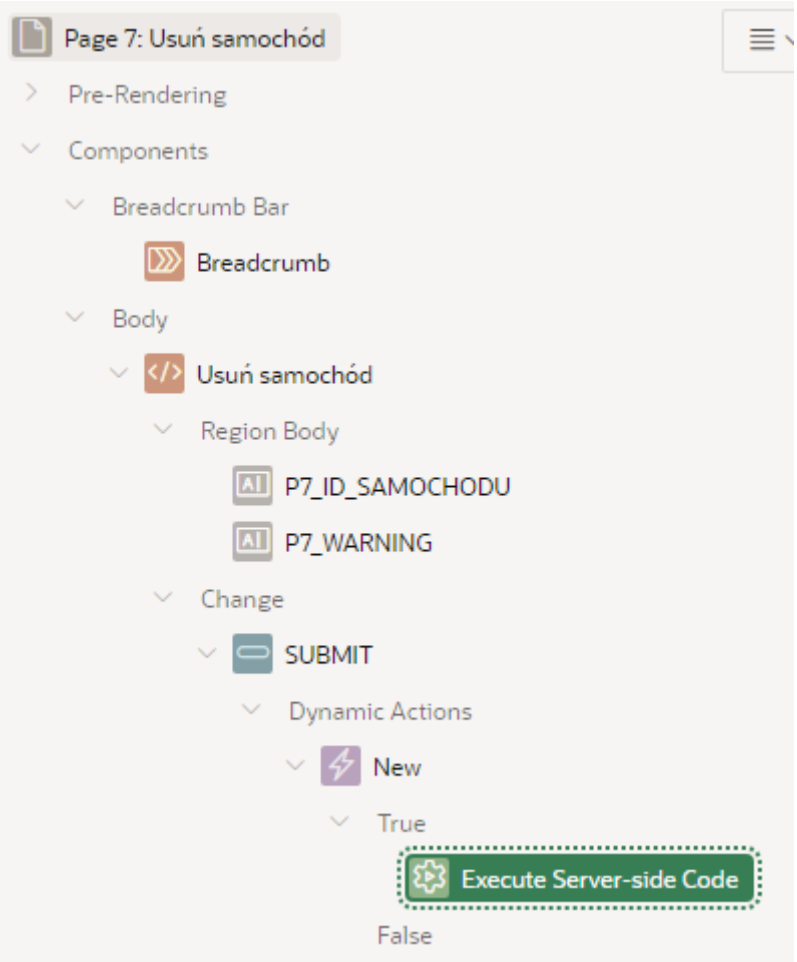
```
1   select * from klienci order by ID_KLIENTA desc
```

Results | Explain | Describe | Saved SQL | History

ID_KLIENTA	NAZWISKO	IMIE	PESEL
24	Krzysik	Daniel	00005000000
23	Leśniak	Paweł	00001100119

7. Usuń samochód

Page Designer:

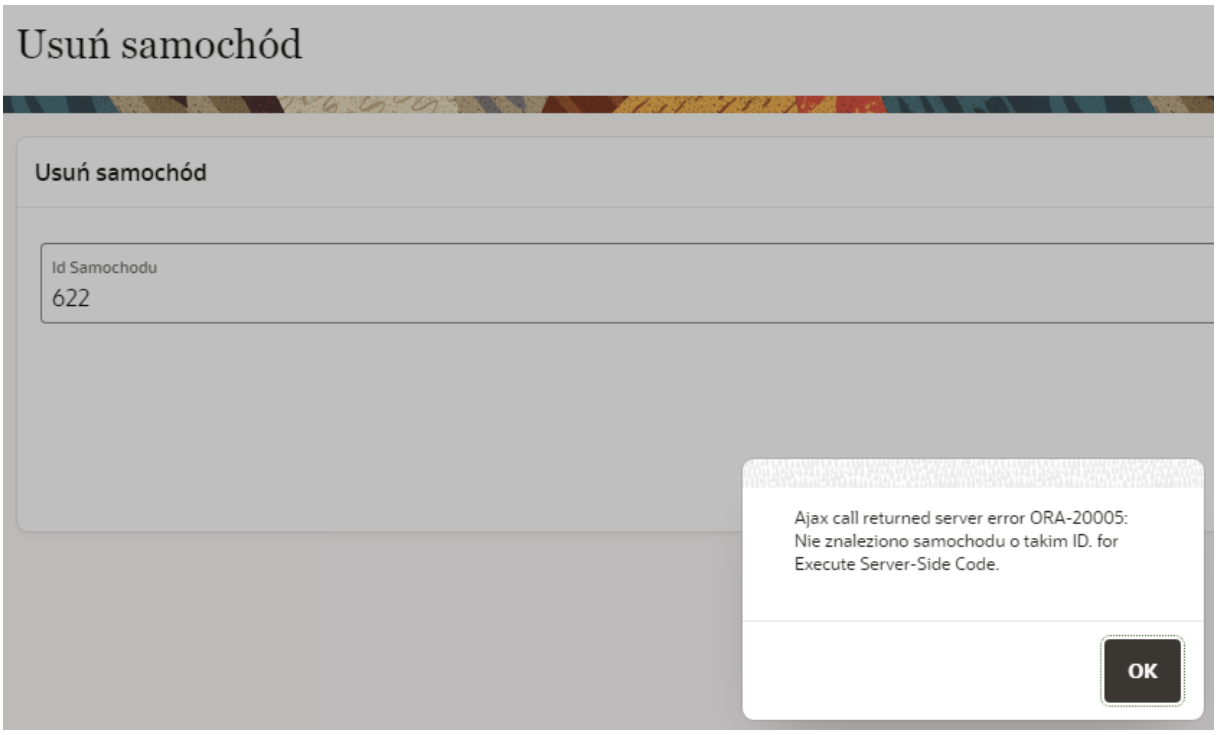


Kod:

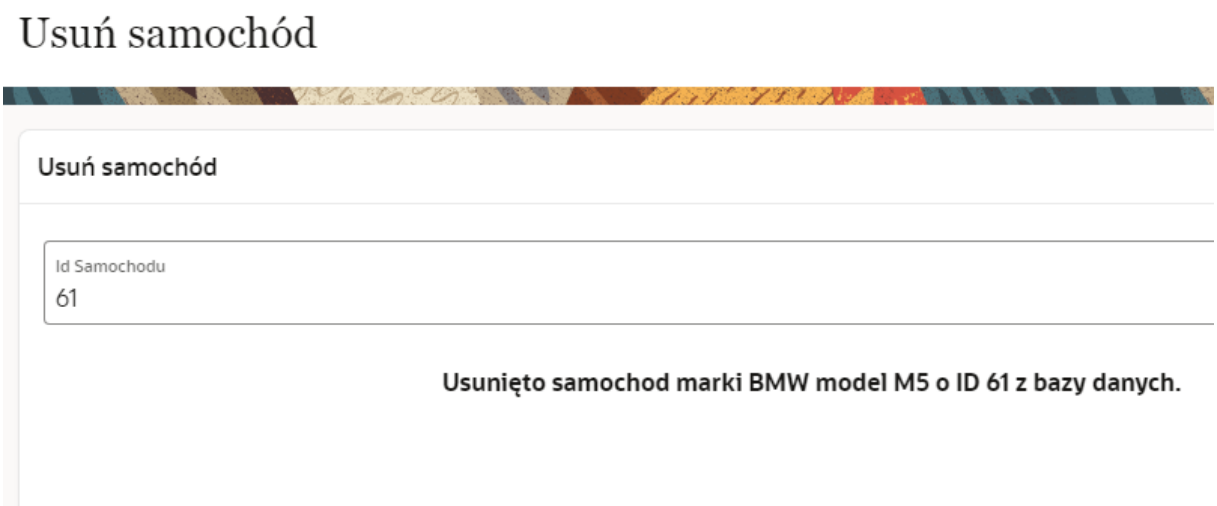
```
pakiet2.usun_samochod(
    :P7_ID_SAMOCHODU,
    :P7_WARNING
    );
```

Wygląd:

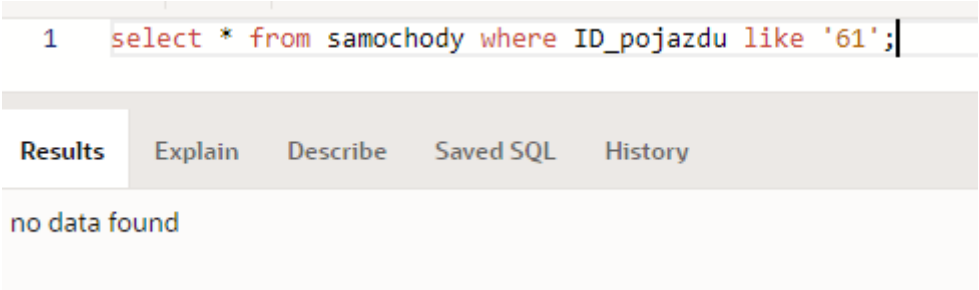
Gdy podamy złe ID:



Poprawione:

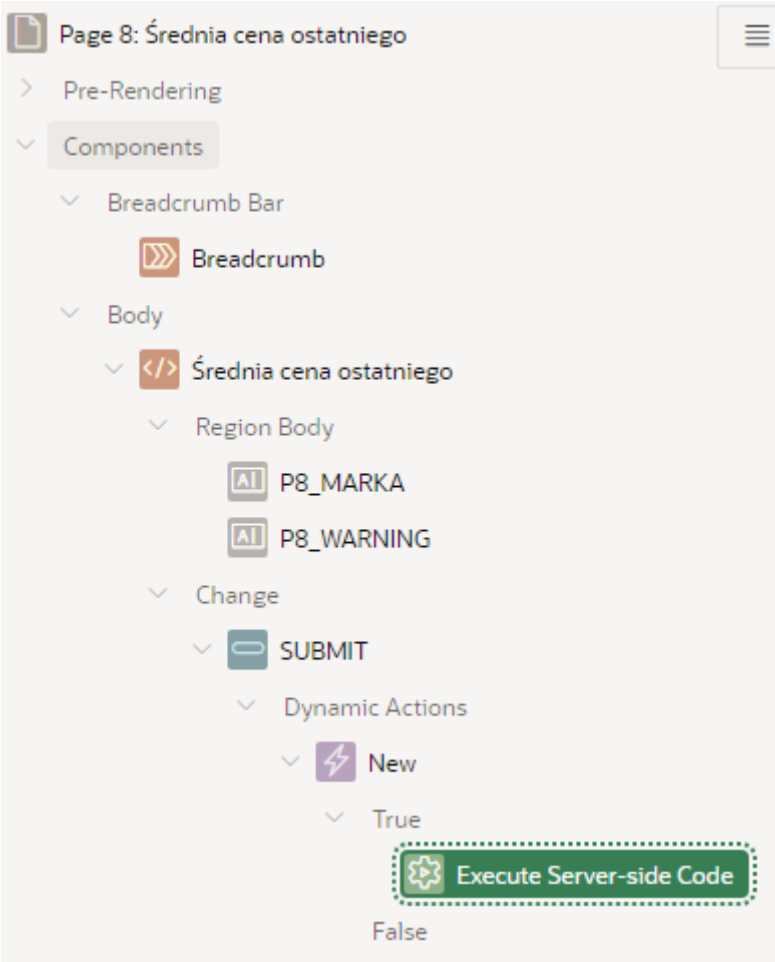


Sprawdzenie w bazie danych:



8. Średnia cena ostatniego

Page Designer:

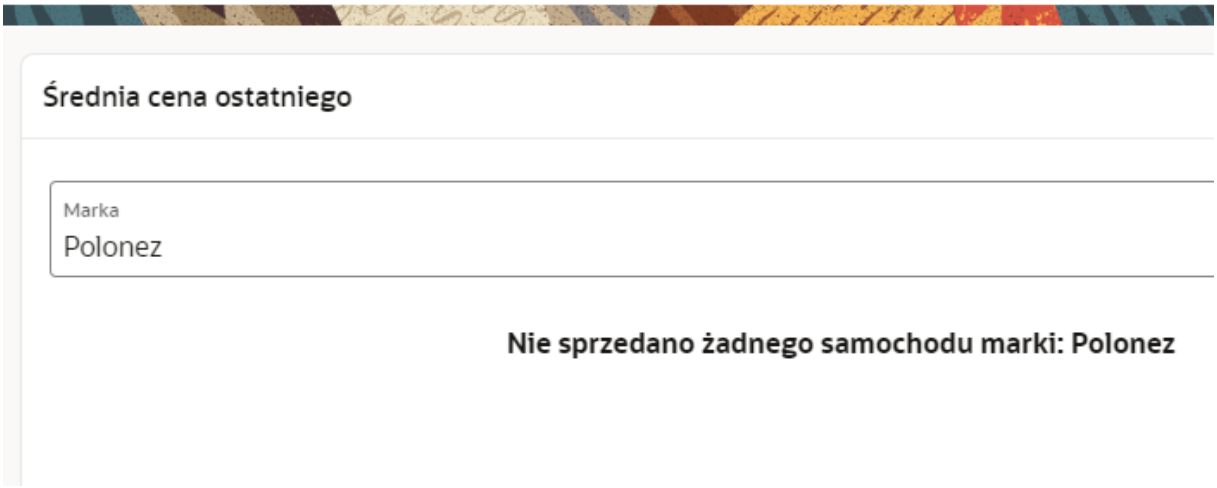


Kod:

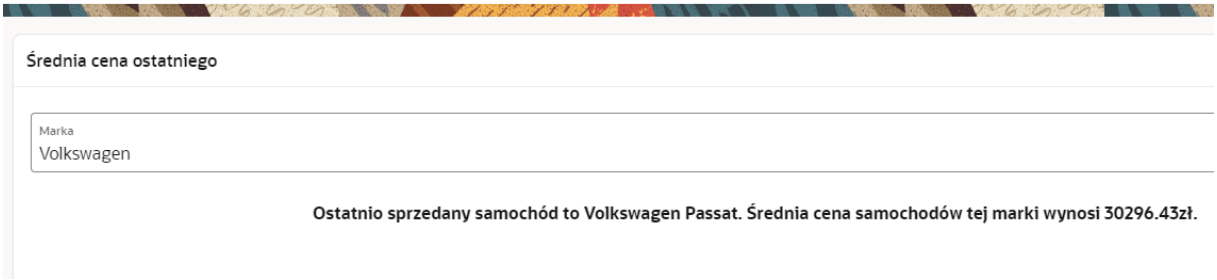
```
pakiet1.SREDNIA_CENA_OSTATNIEGO(
    :P8_MARKA,
    :P8_WARNING);
```

Wygląd:

Średnia cena ostatniego

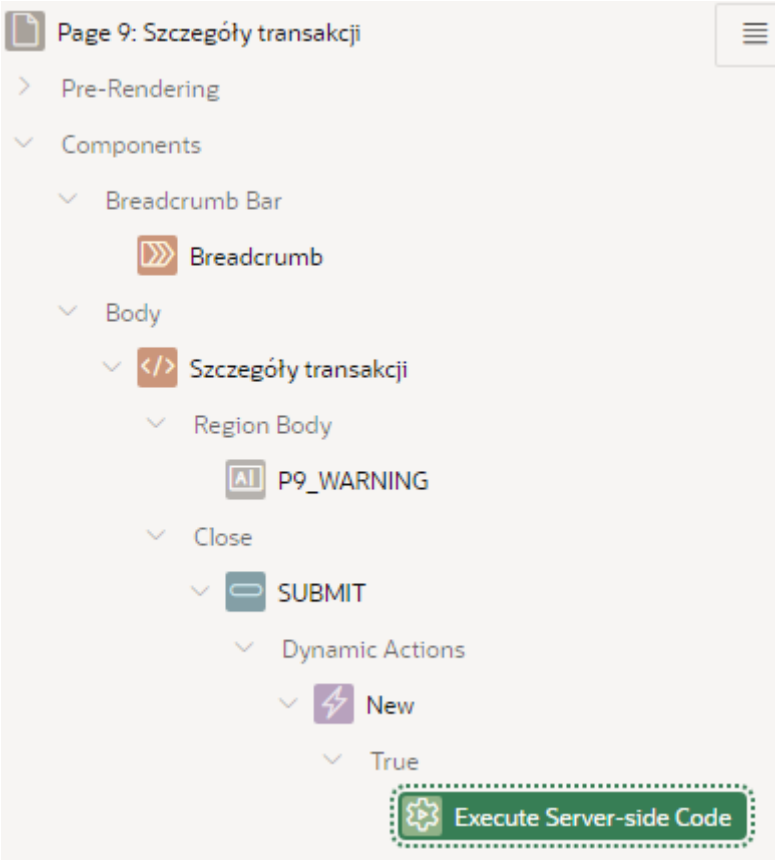


Średnia cena ostatniego



9. Szczegóły ostatniej transakcji

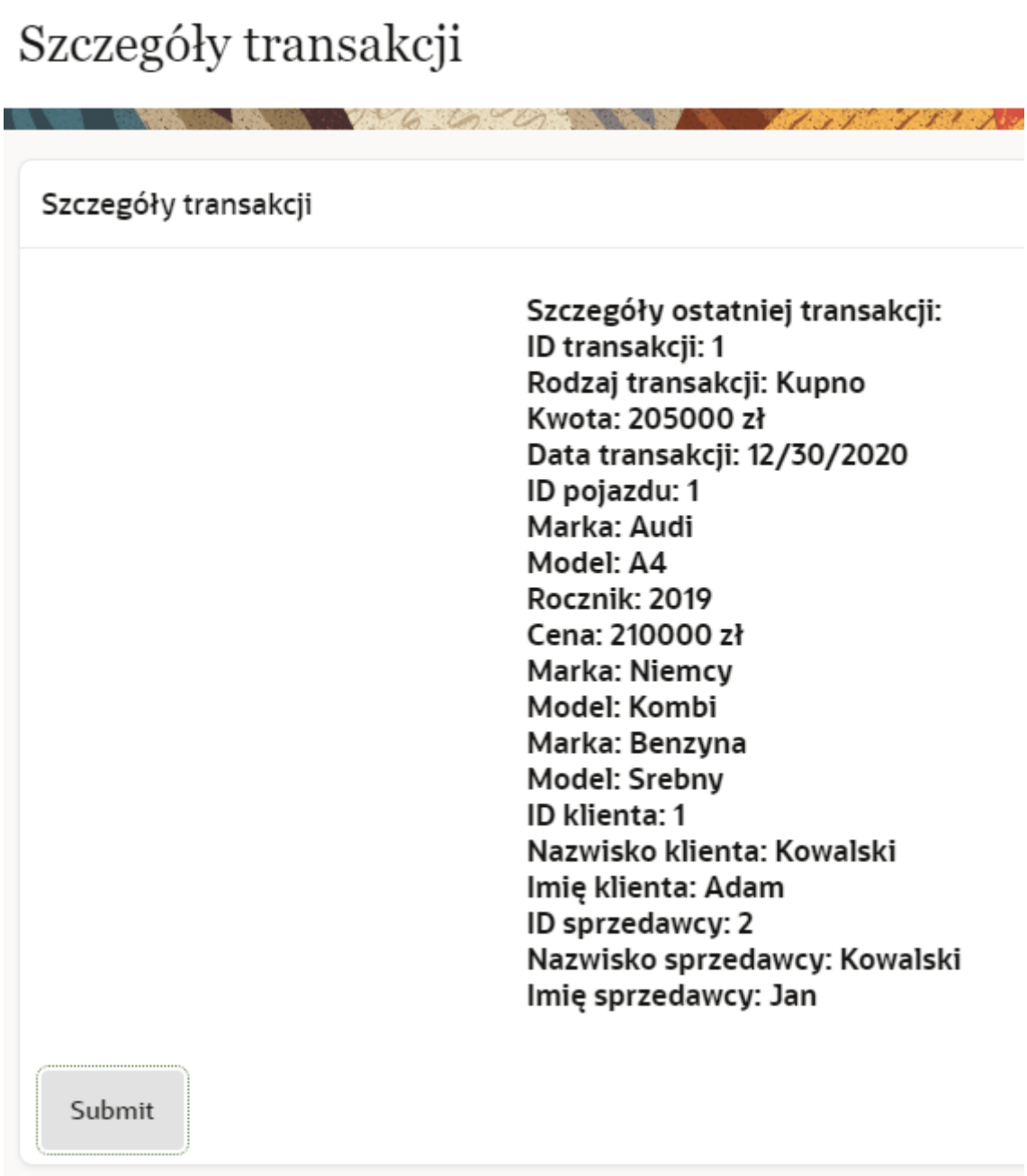
Page Designer:



Kod:

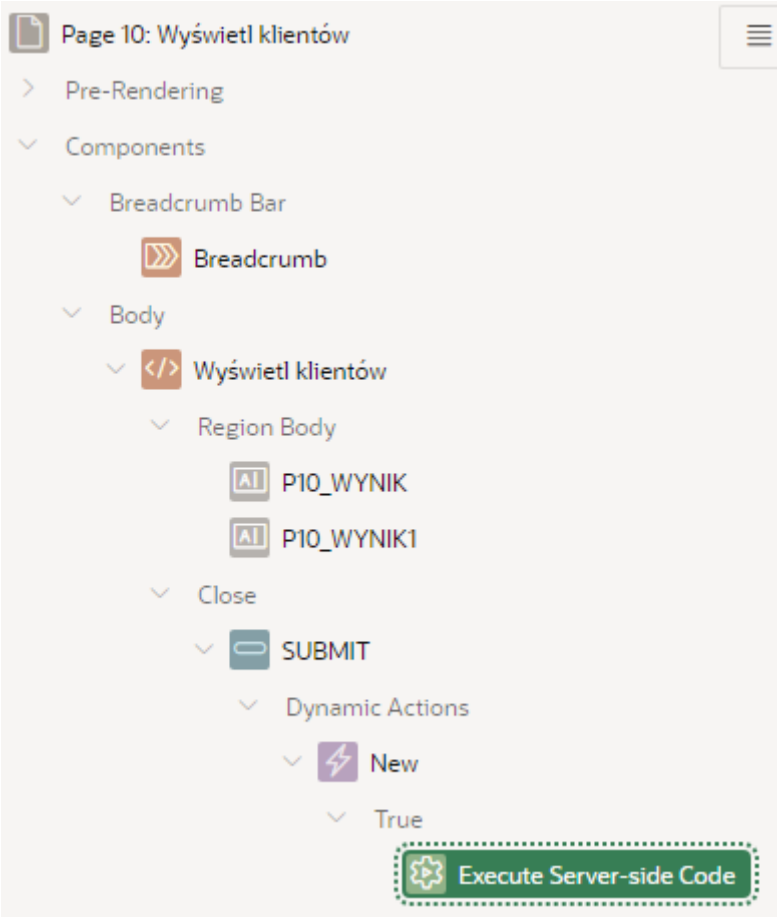
```
pakiet2.SZCZEGOLY_TRANSAKCJI(
    :P9_WARNING
);
```

Wygląd:



10. Wyświetl klientów

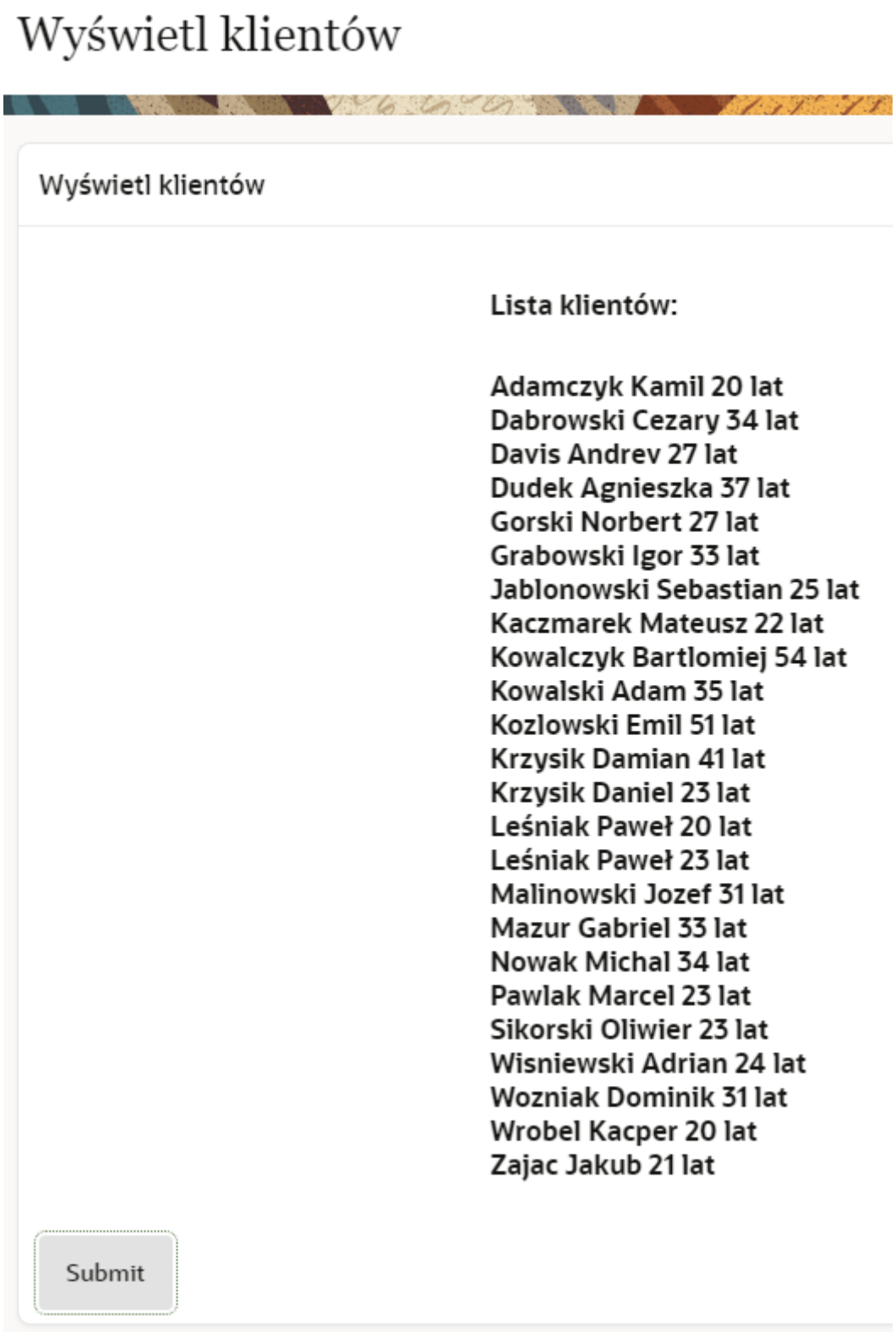
Page Designer:



Kod:

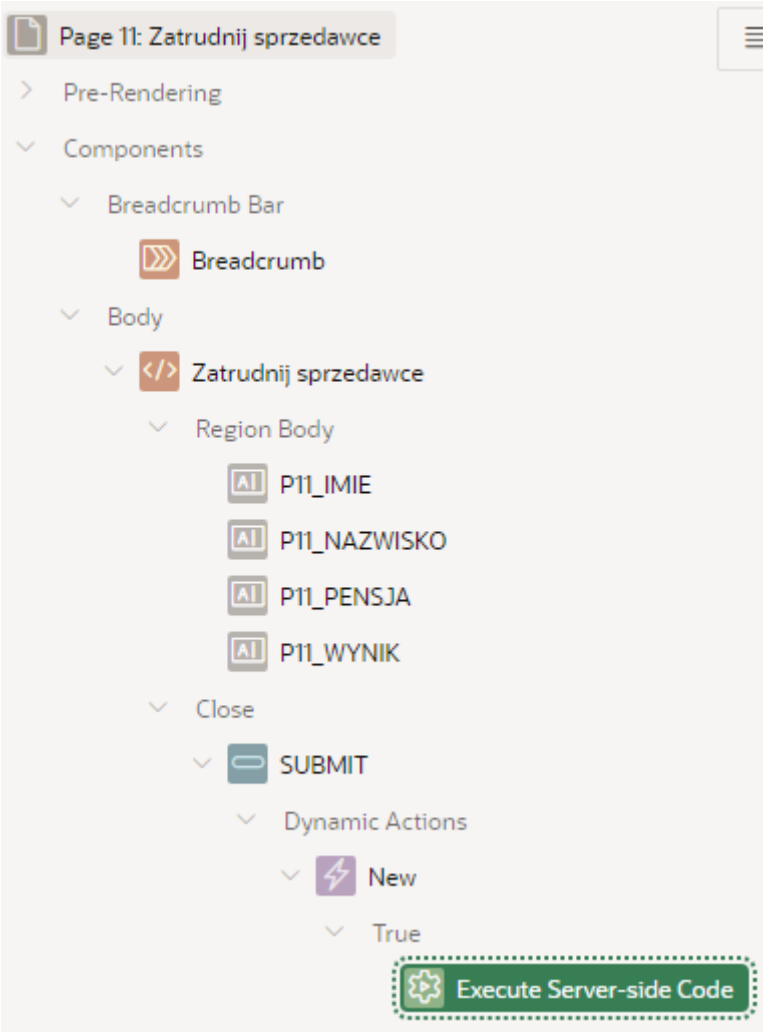
```
pakiet2.WYSWIETL_KLIENTOW(
    :P10_WYNIK,
    :P10_WYNIK1
);
```

Wygląd:



11. Zatrudnij sprzedawcę

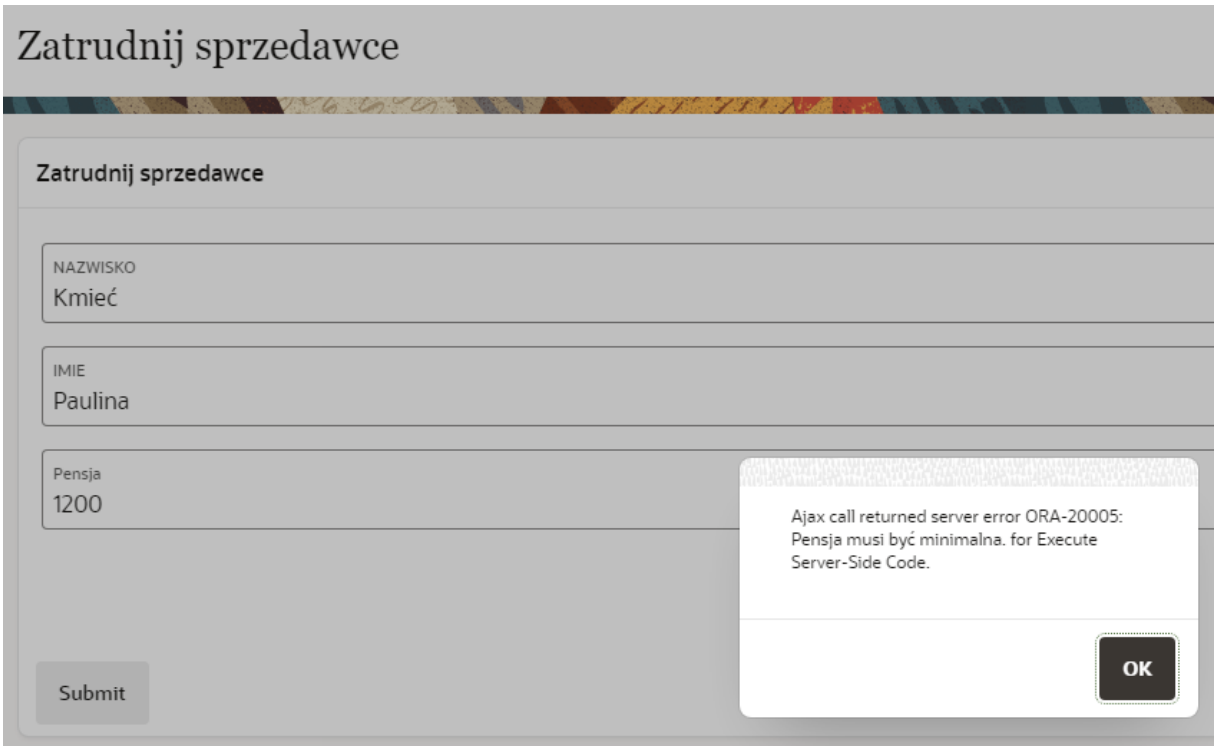
Page Designer:



Kod:

```
pakiet3.ZATRUDNIJ_SPRZEDAWCE(
:P11_NAZWISKO,
:P11_IMIE,
:P11_PENSJA,
:P11_WYNIK
);
```

Wygląd:



Sprawdzenie czy jest w bazie:

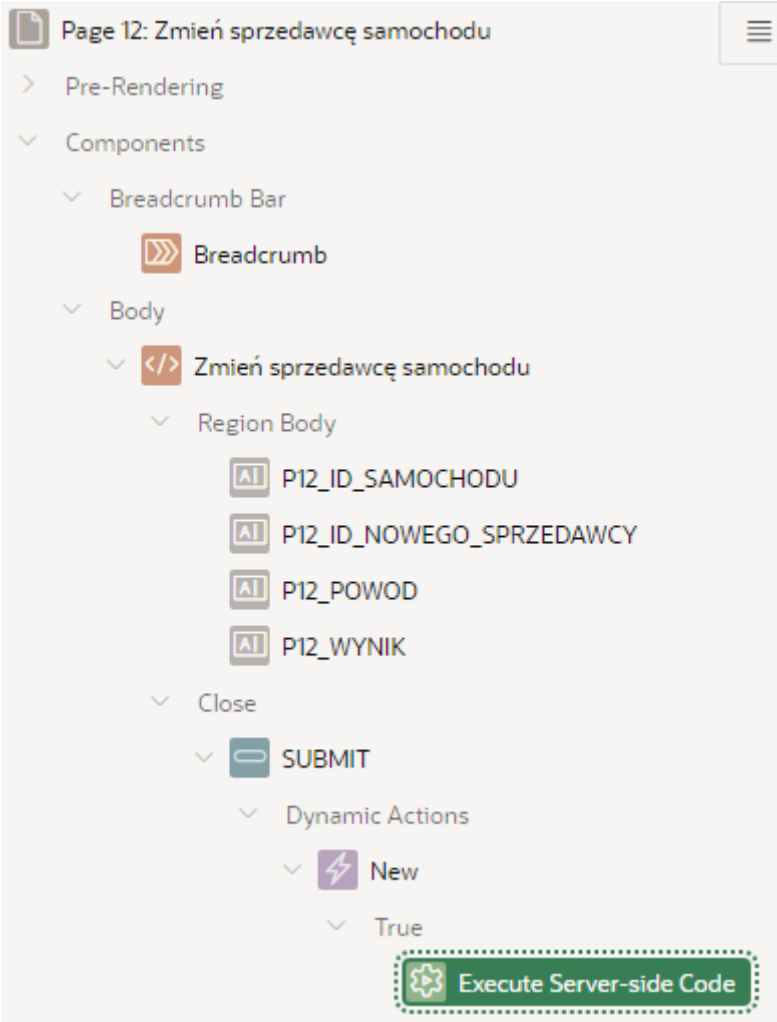
```
1  select * from sprzedawcy where nazwisko like 'Kmieć';
```

Results | Explain | Describe | Saved SQL | History

ID_SPRZEDAWCY	NAZWISKO	IMIE
10	Kmieć	Paulina

12. Zmień sprzedawcę samochodu

Page Designer:



Kod:

```
1  pakiet3.ZMIEN_SPRZEDAWCE_SAMOCHODU(
2  :P12_ID_SAMOCHODU,
3  :P12_ID_NOWEGO_SPRZEDAWCY,
4  :P12_POWOD,
5  :P12_WYNIK);
```

Wygląd:

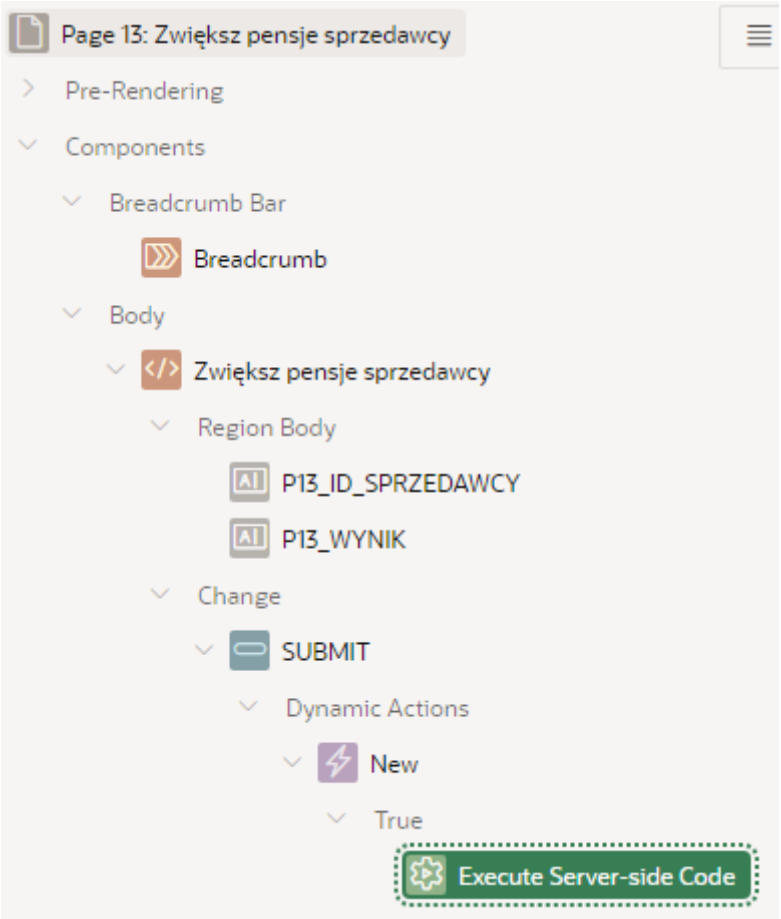
Zmień sprzedawcę samochodu
Zmień sprzedawcę samochodu
Id Samochodu
266
Id Nowego Sprzedawcy
11
Powód
Pomyłka przy wpisaniu
Ajax call returned server error ORA-20005: W bazie danych nie ma takiego samochodu. for Execute Server-Side Code.
OK
Submit

Zmień sprzedawcę samochodu
Zmień sprzedawcę samochodu
Id Samochodu
26
Id Nowego Sprzedawcy
11
Powód
Pomyłka przy wpisaniu
Ajax call returned server error ORA-20002: W bazie danych nie ma takiego sprzedawcy. for Execute Server-Side Code.
OK
Submit

Zmień sprzedawcę samochodu

13. Zwiększ pensje sprzedawcy

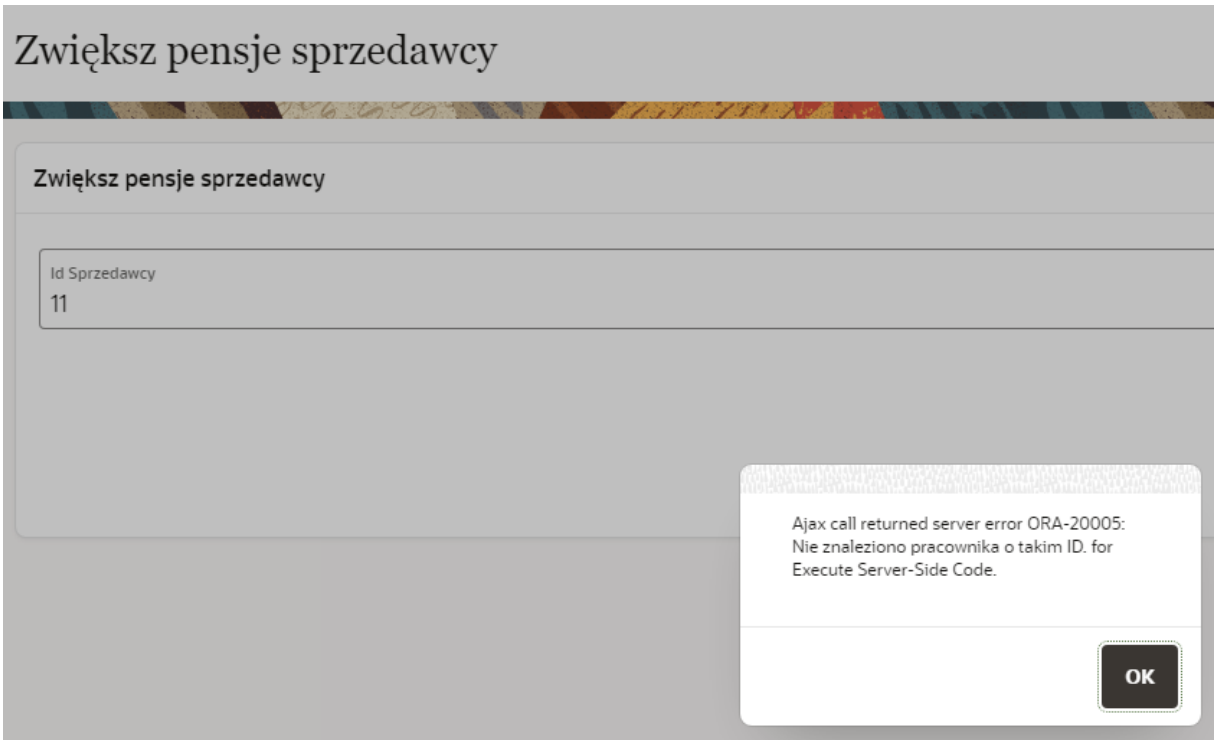
Page Designer:

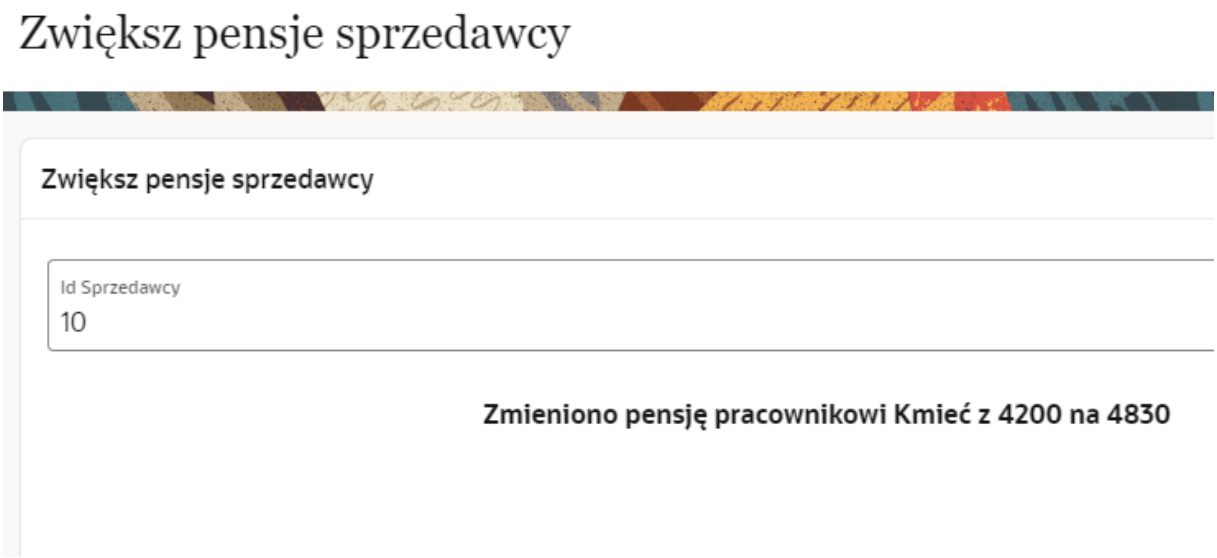


Kod:

```
pakiet3.ZWIEKSZPENSJESPRZEDAWCY(
:P13_ID_SPRZEDAWCY,
:P13_WYNIK);
```

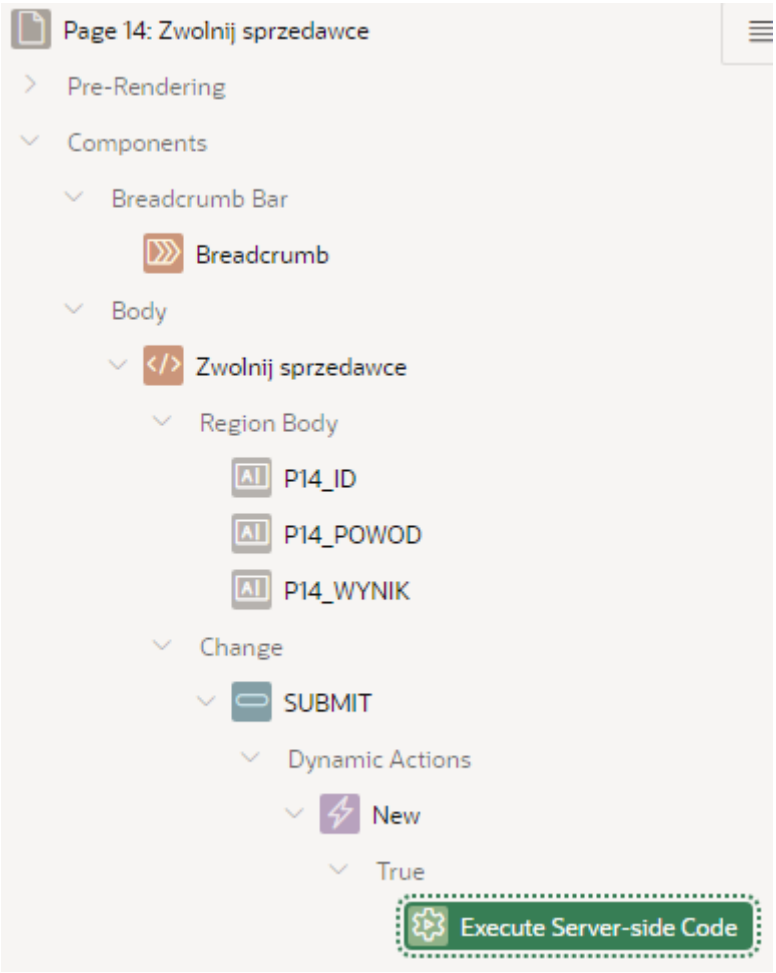
Wygląd:





14. Zwolnij sprzedawcę

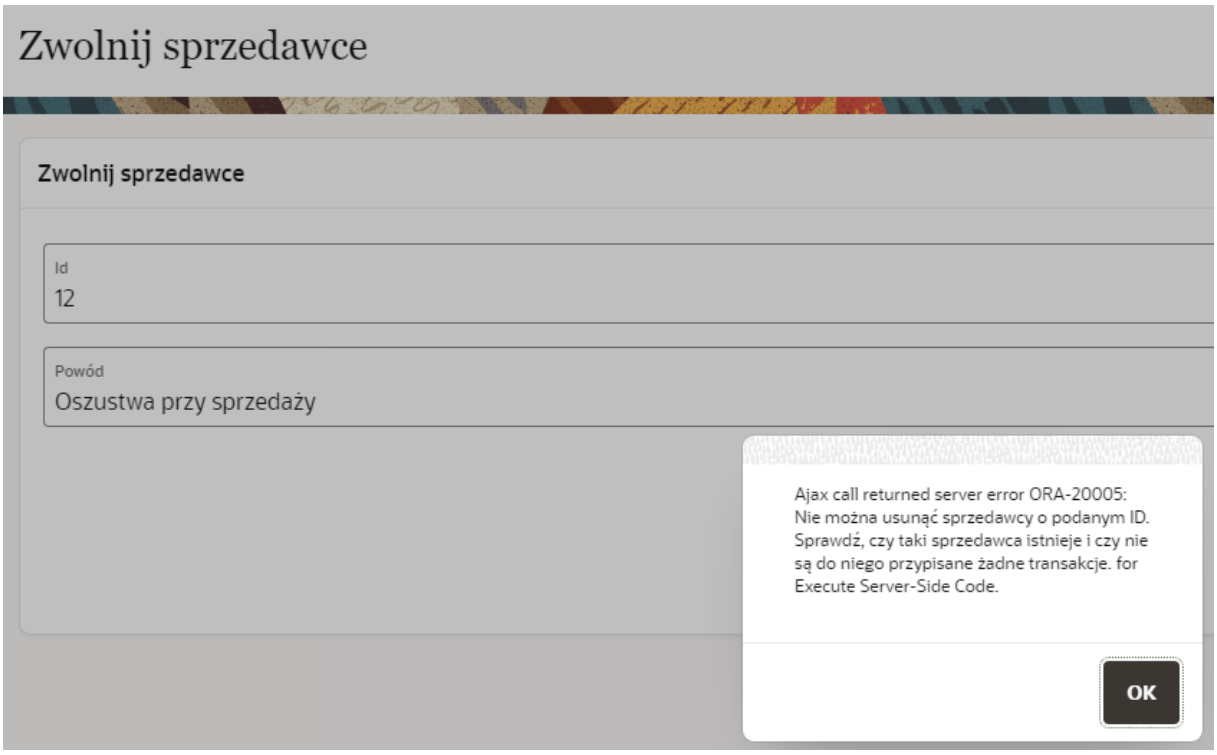
Page Designer:



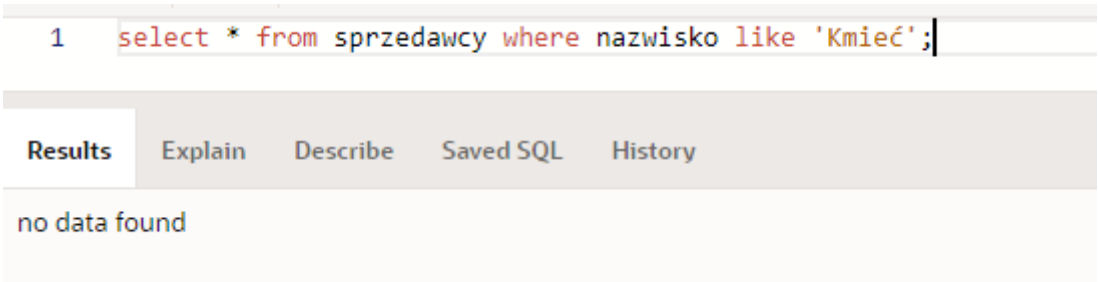
Kod:

```
pakiet3.ZWOLNIJSPRZEDAWCE(
:P14_ID,
:P14_POWOD,
:P14_WYNIK);
```

Wygląd:

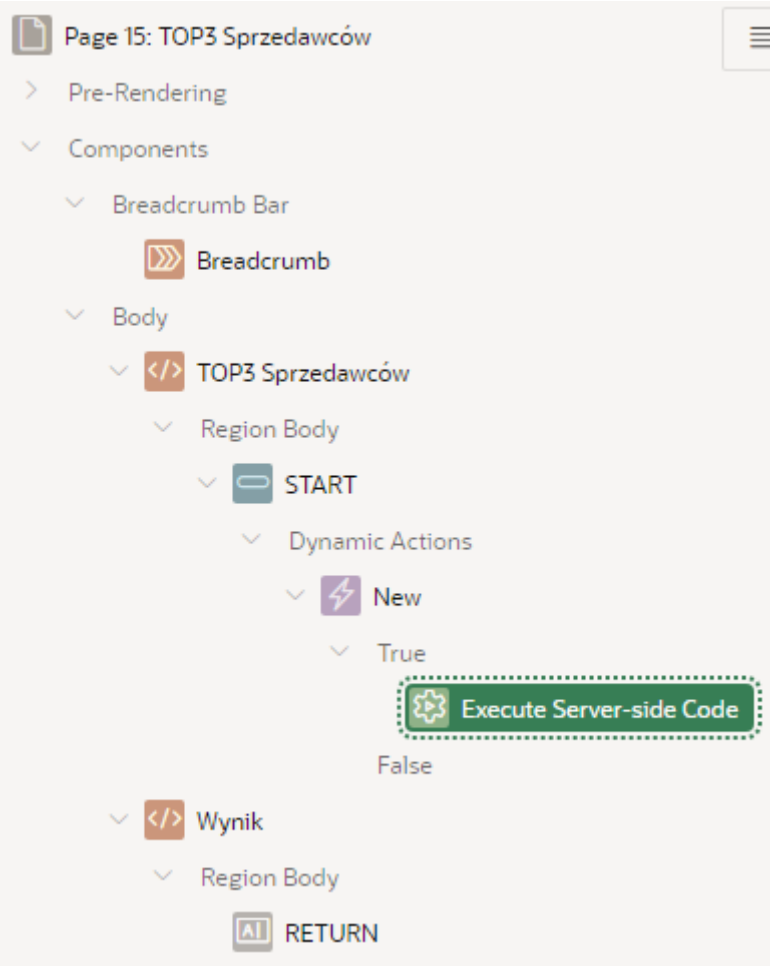


Sprawdzenie w bazie:



15. Top3 sprzedawców

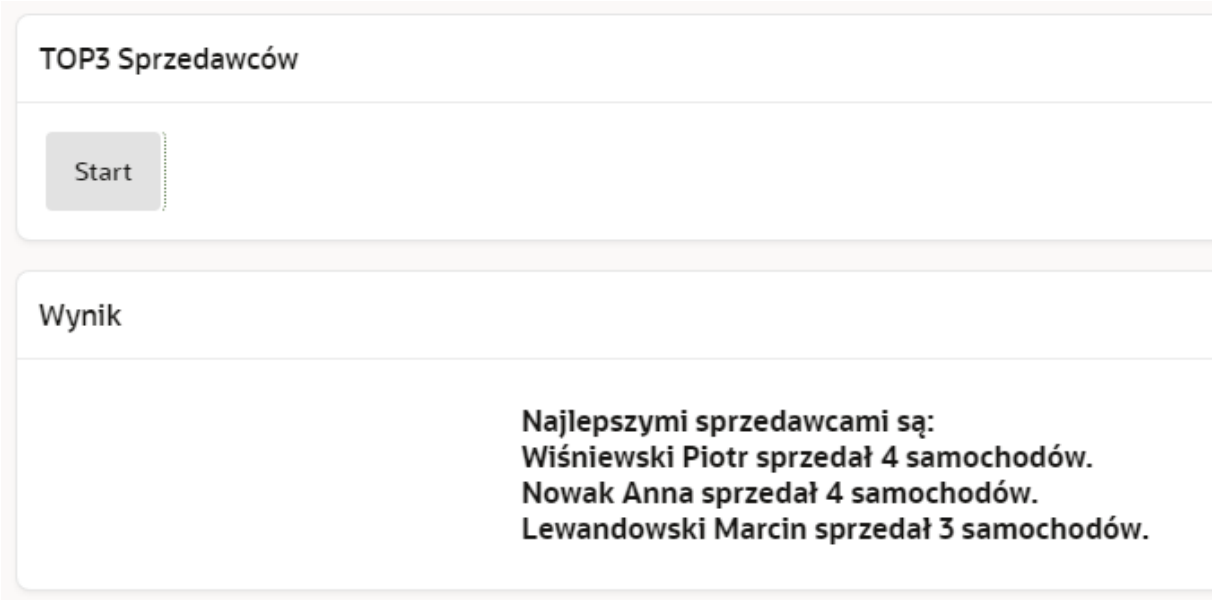
Page Designer:



Kod:

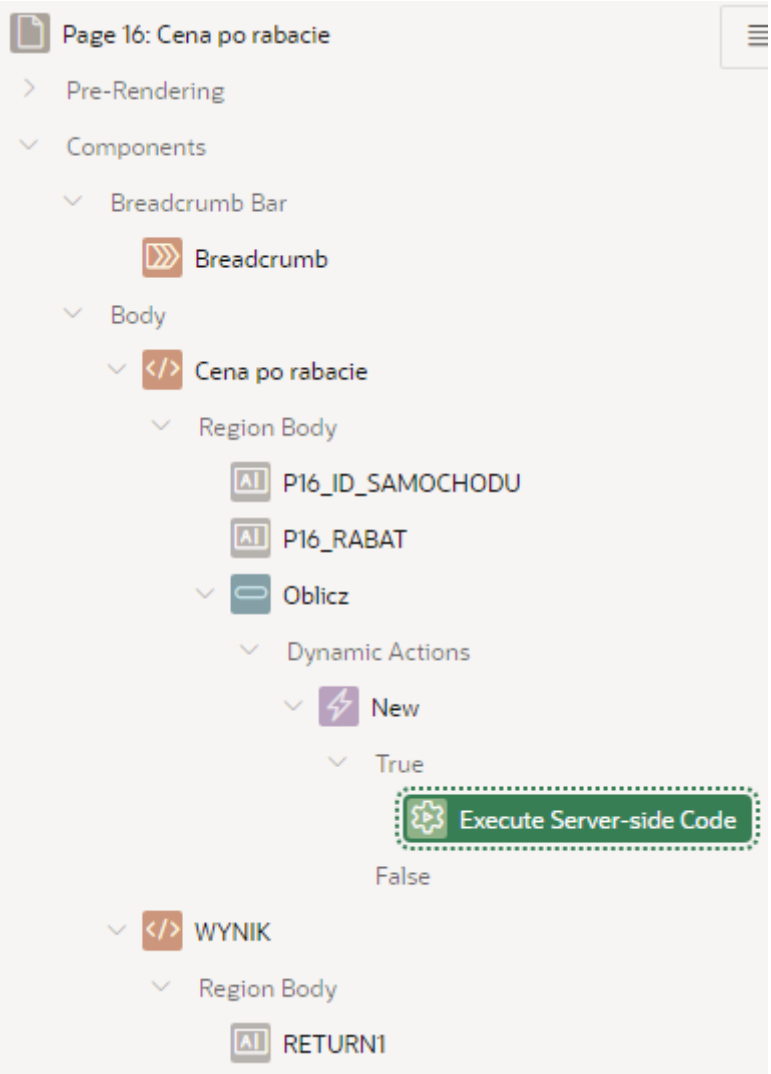
```
BEGIN
    :RETURN :=
pakiet3.top_3_najlepszych_sprzedawcow();
END;
```

Wygląd:



16. Cena po rabacie

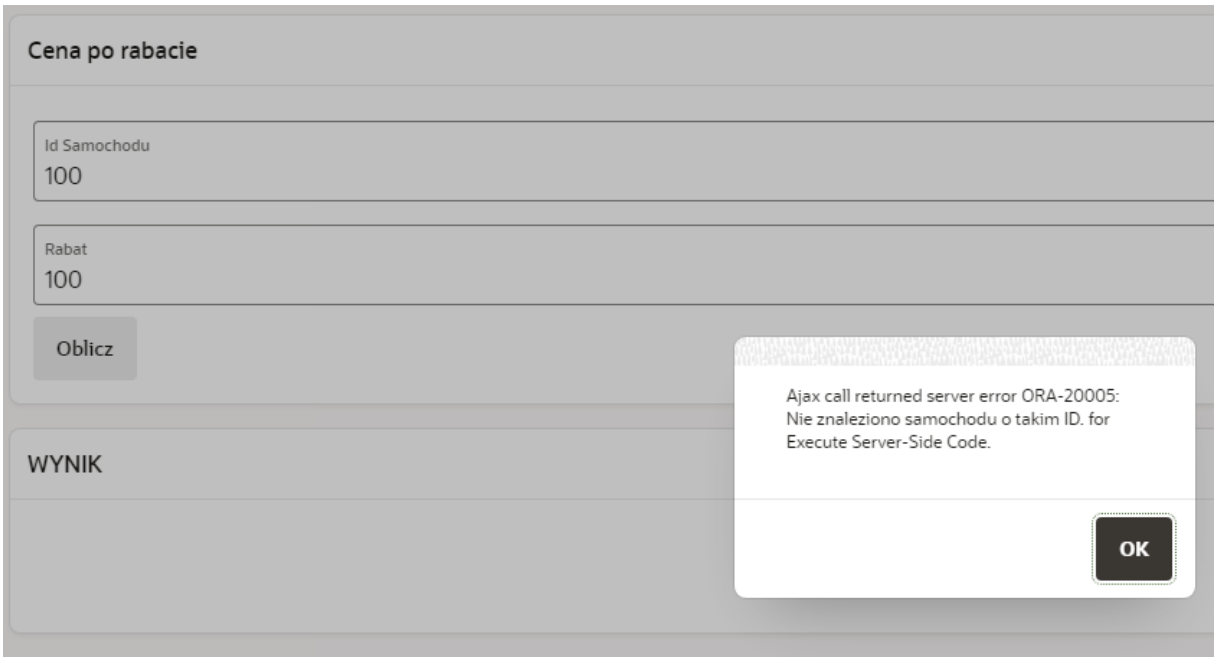
Page Designer:

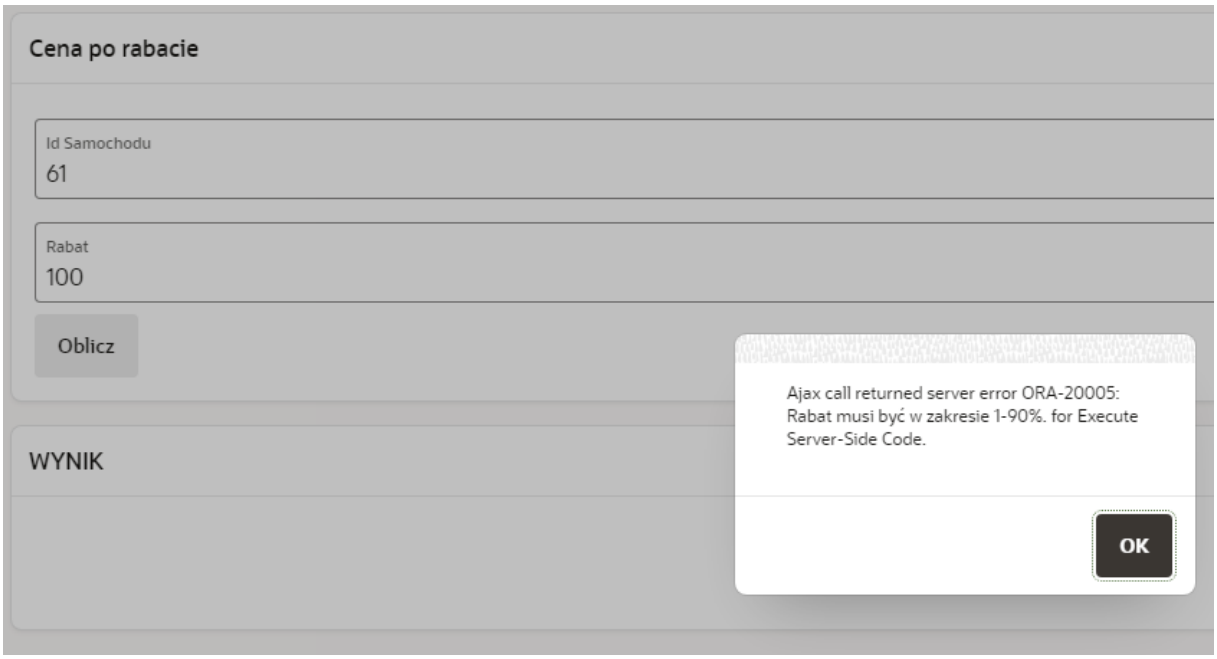


Kod:

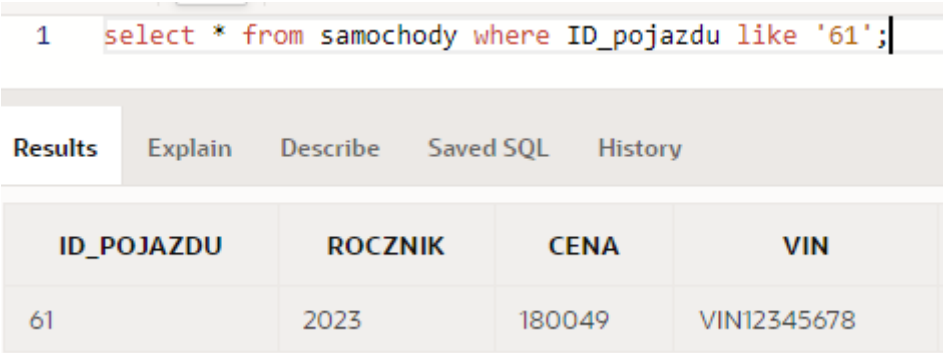
```
BEGIN
    :RETURN1 :=
pakiet1.cena_z_rabatem(
:P16_ID_SAMOCHODU,
:P16_RABAT
);
END;
```

Wygląd:



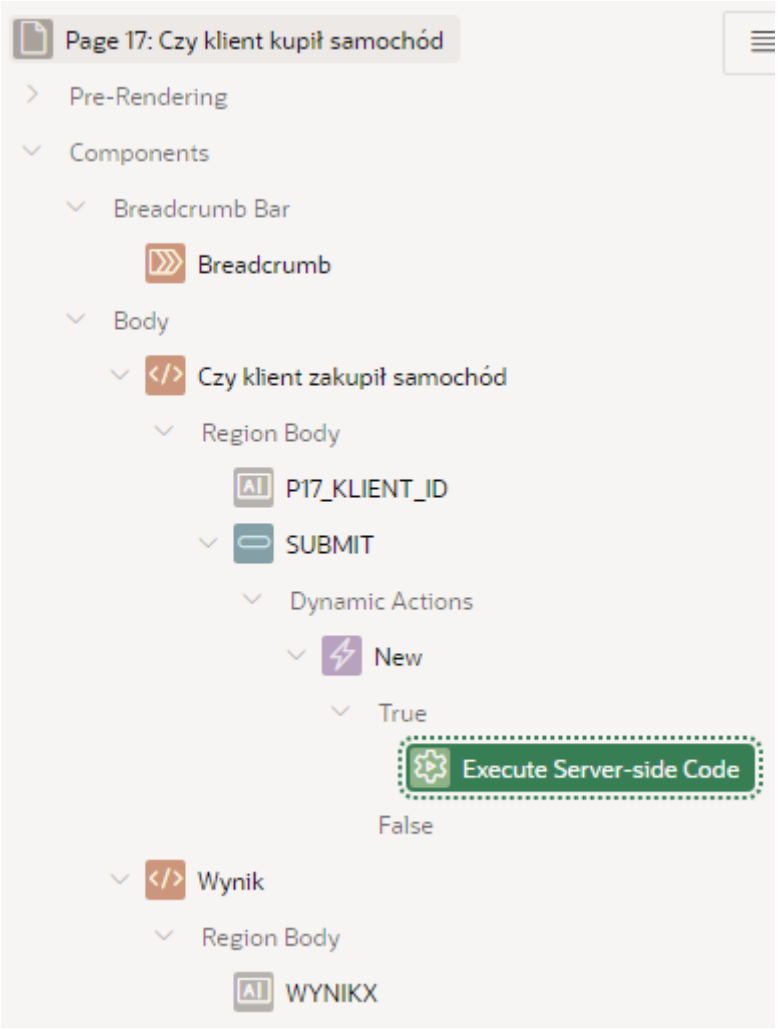


Sprawdzenie w bazie:



17. Czy klient zakupił samochód

Page Designer:



Kod:

```
BEGIN
    :WYNIKX :=
pakiet1.czy_klient_kupil_samochod(
:P17_KLIENT_ID
);
END;
```

Wygląd:

Czy klient zakupił samochód

Klient Id
25

Submit

Wynik

W bazie danych nie ma takiego klienta.

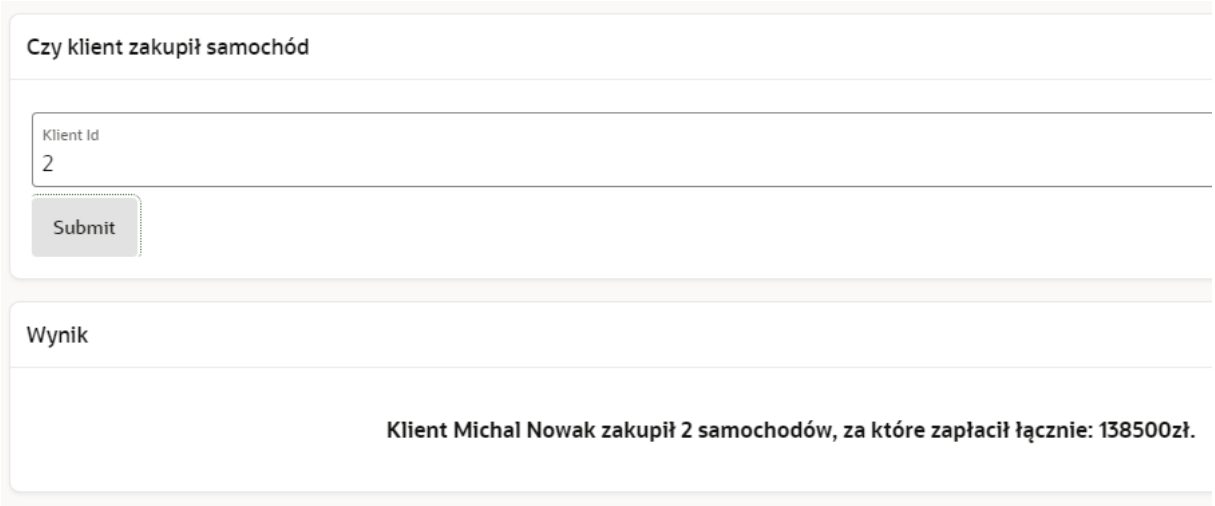
Czy klient zakupił samochód

Klient Id
23

Submit

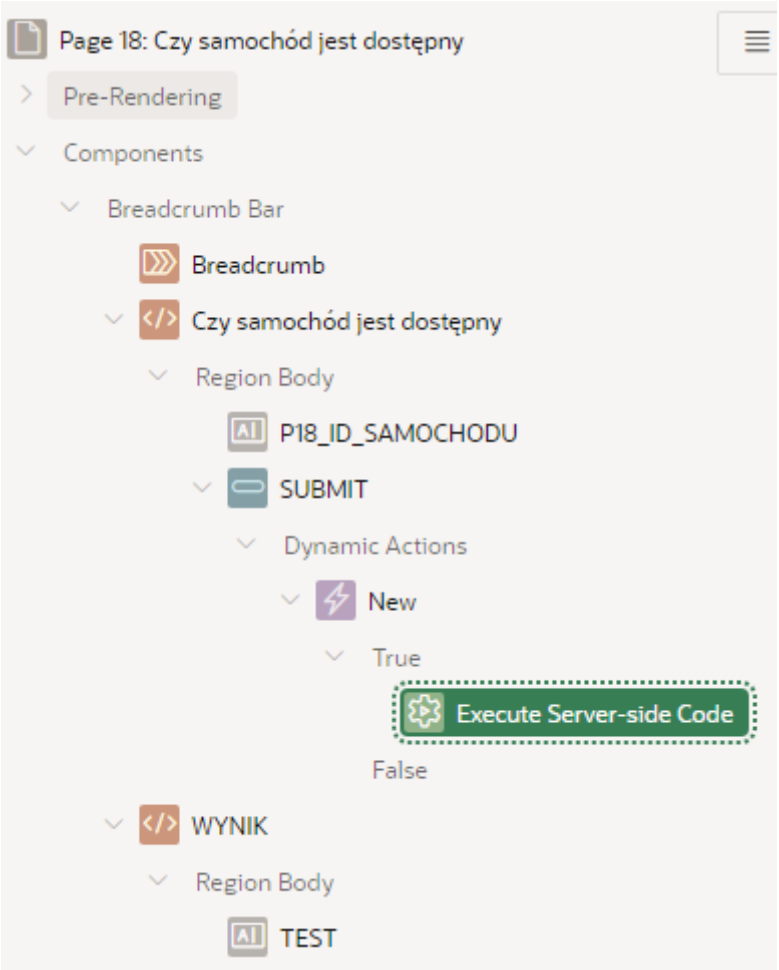
Wynik

Klient Daniel Krzysik nie zakupił żadnego samochodu.



18. Czy samochód jest dostępny

Page Designer:



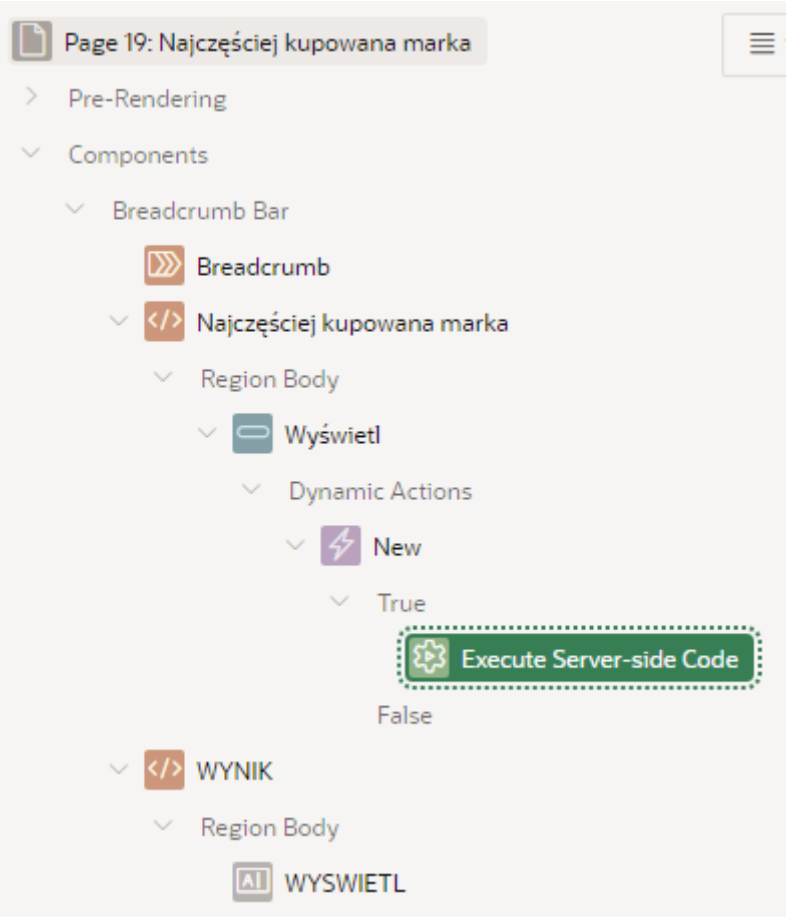
Kod:

```
BEGIN
   :TEST :=
pakiet1.czy_samochod_dostepny(
:P18_ID_SAMOCHODU
);
END;
```

Wygląd:

19. Najczęściej kupowana marka

Page Designer:



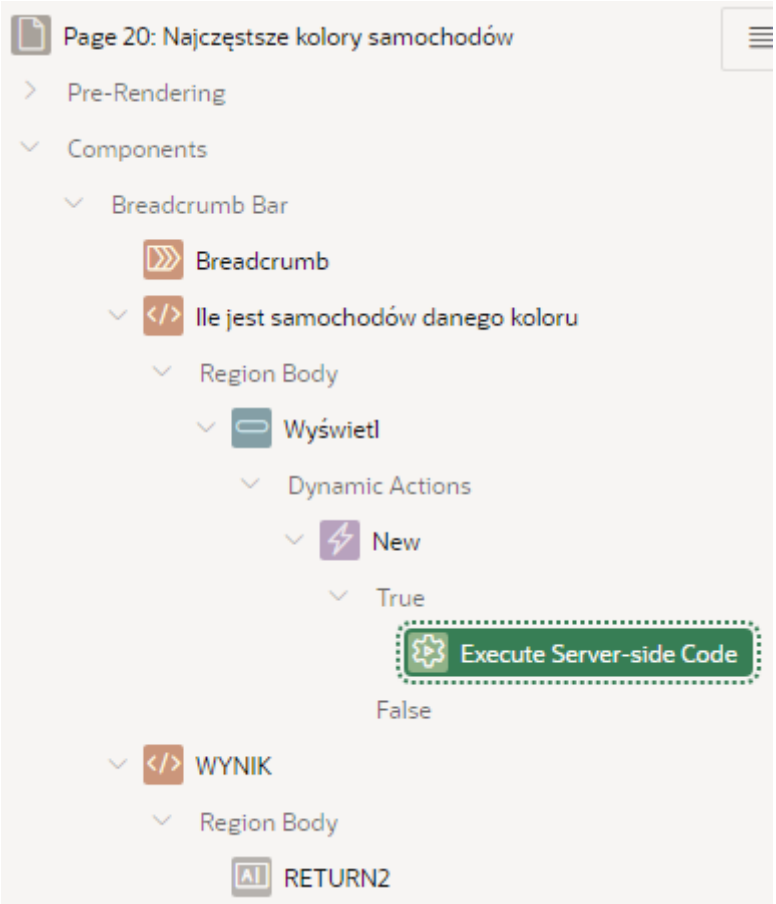
Kod:

```
BEGIN
   :WYSWIETL :=
pakiet1.najczesciej_kupowana_marka(
);
END;
```

Wygląd:

20. Najczęstsze kolory samochodów

Page Designer:



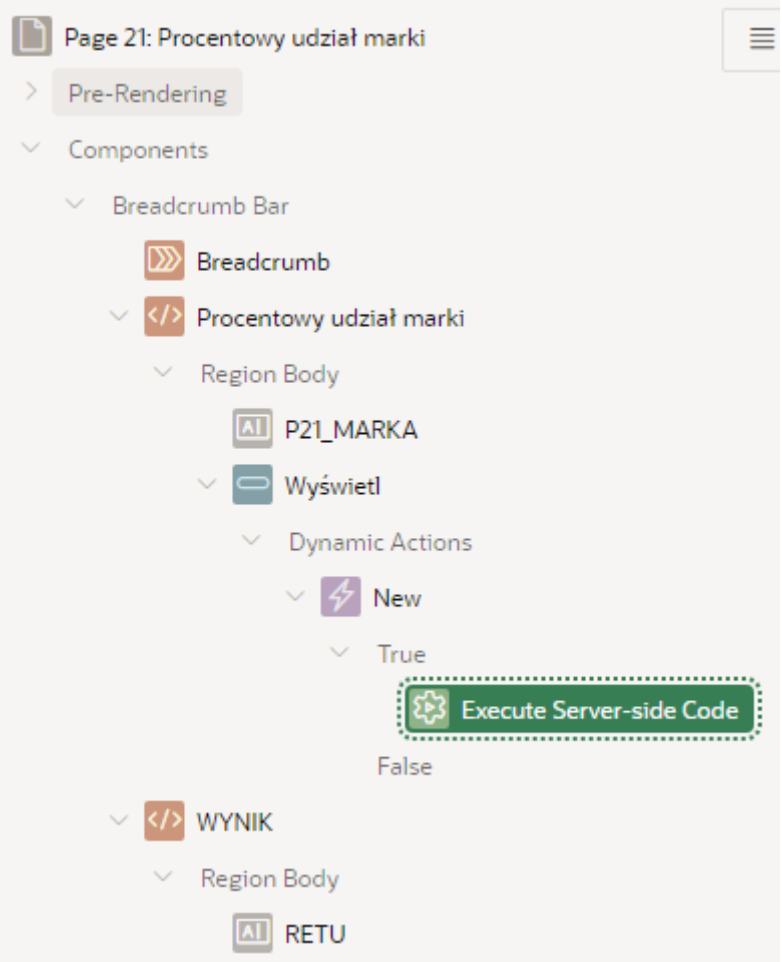
Kod:

```
BEGIN
    :RETURN2 :=
pakiet2.najczestsze_kolory_samochodow(
);
END;
```

Wygląd:

21. Procentowy udział marki

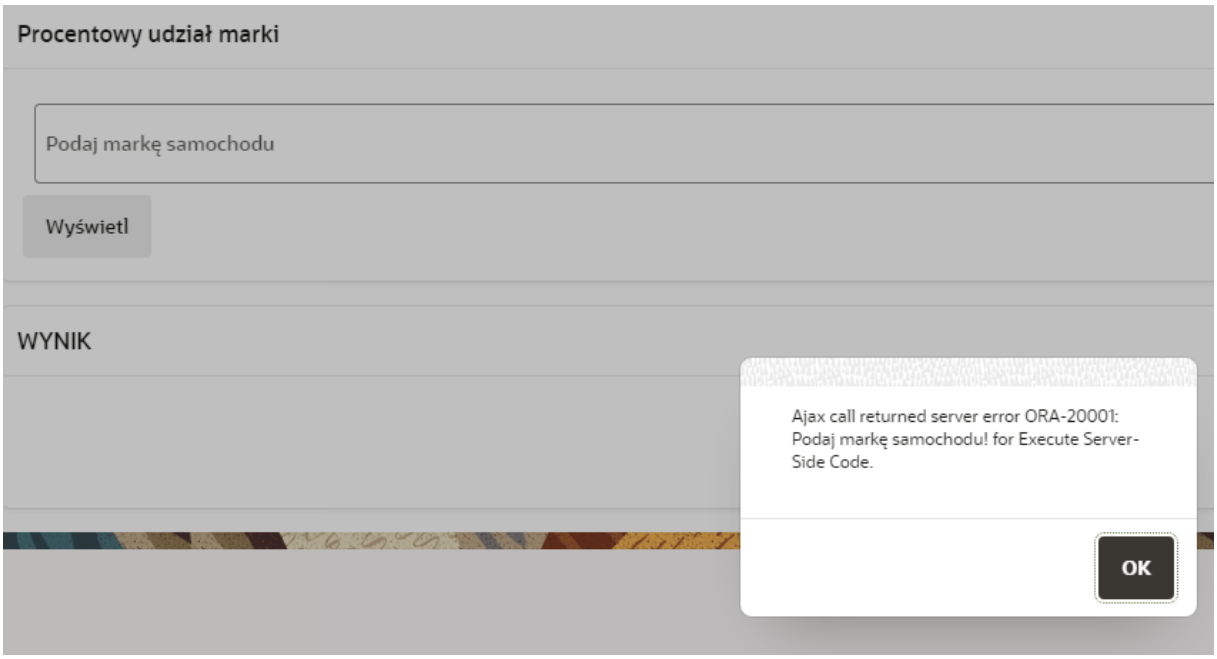
Page Designer:

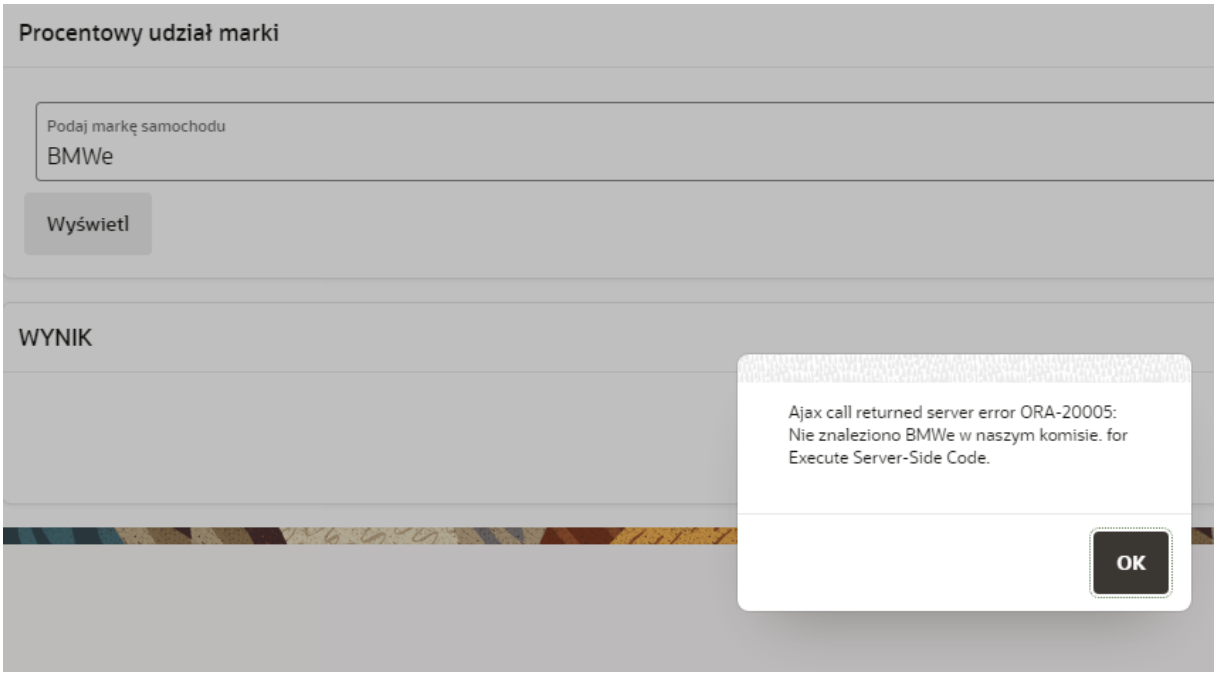


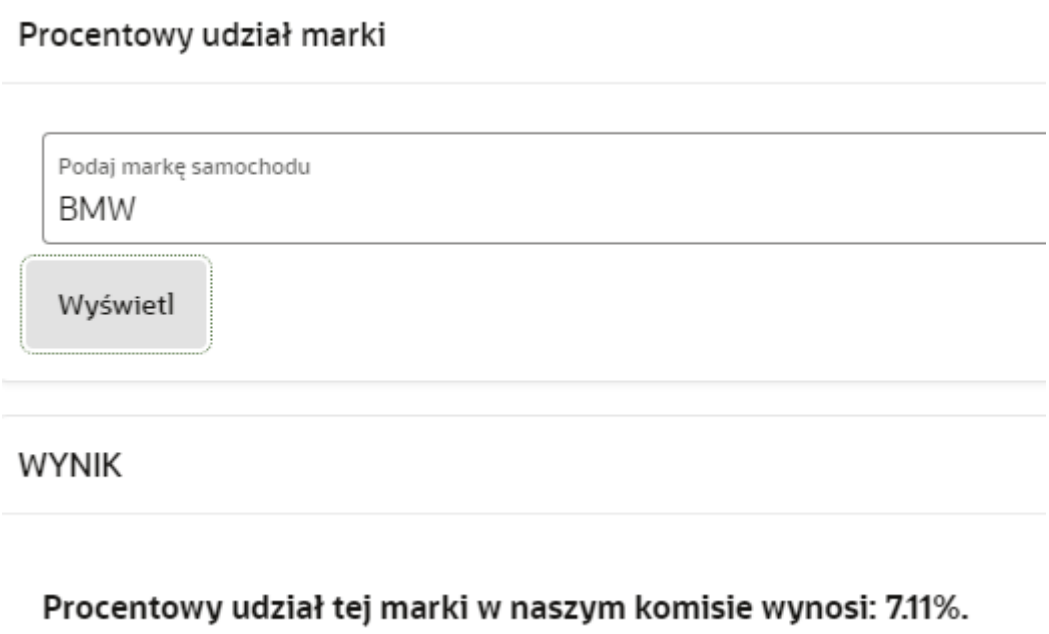
Kod:

```
BEGIN
    :RETU :=
pakiet2.procentowy_udzial_marki(:P21_MARKA
);
END;
```

Wygląd:

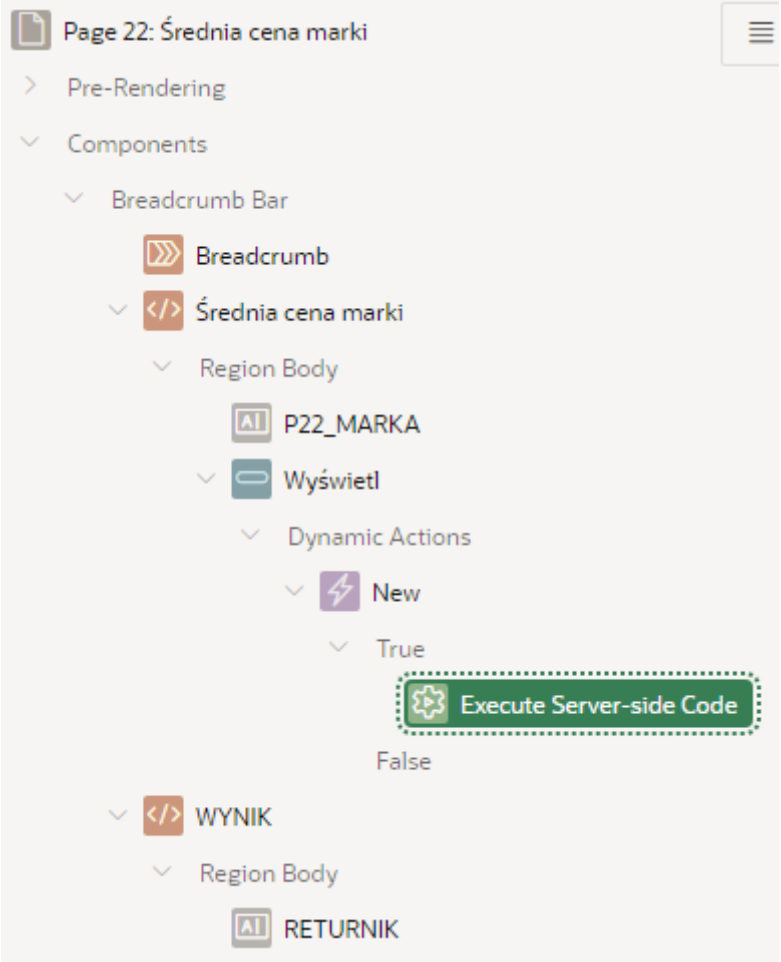






22. Średnia cena marki

Page Designer:



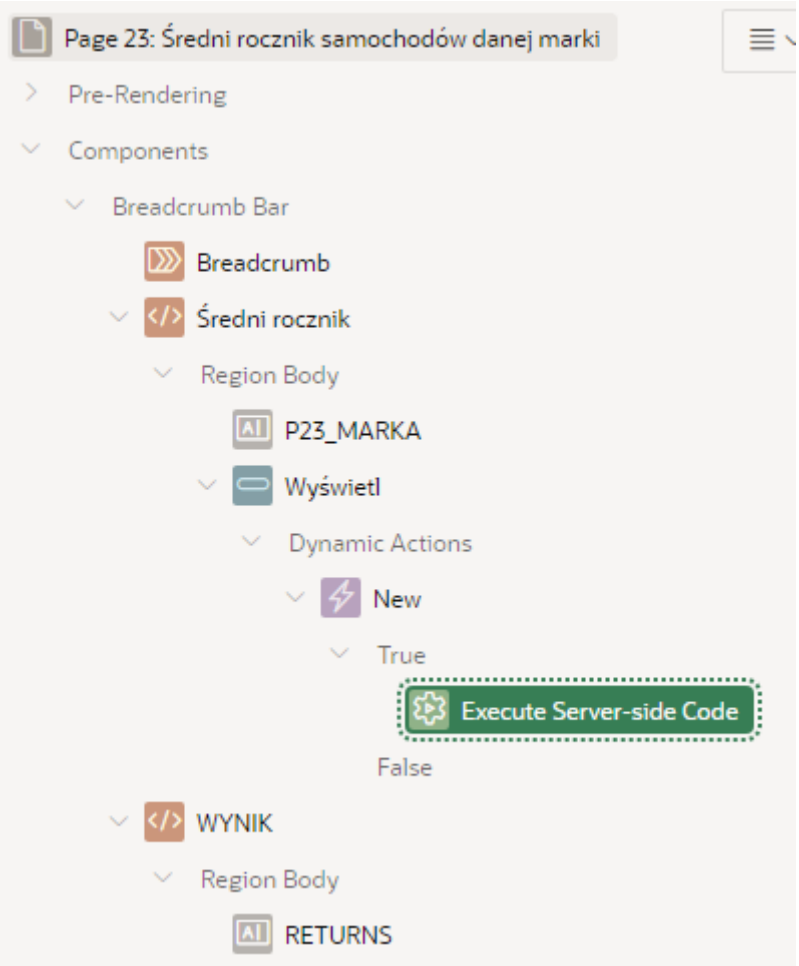
Kod:

```
BEGIN
    :RETURNIK :=
pakiet2.SREDNIA_CENA_MARKI(:P22_MARKA
);
END;
```

Wygląd:

23. Średni rocznik samochodów danej marki

Page Designer:



Kod:

```
BEGIN
   :RETURNS :=
pakiet2.sredni_rocznik_samochodow_danej_marki(:P23_MARKA
);
END;
```

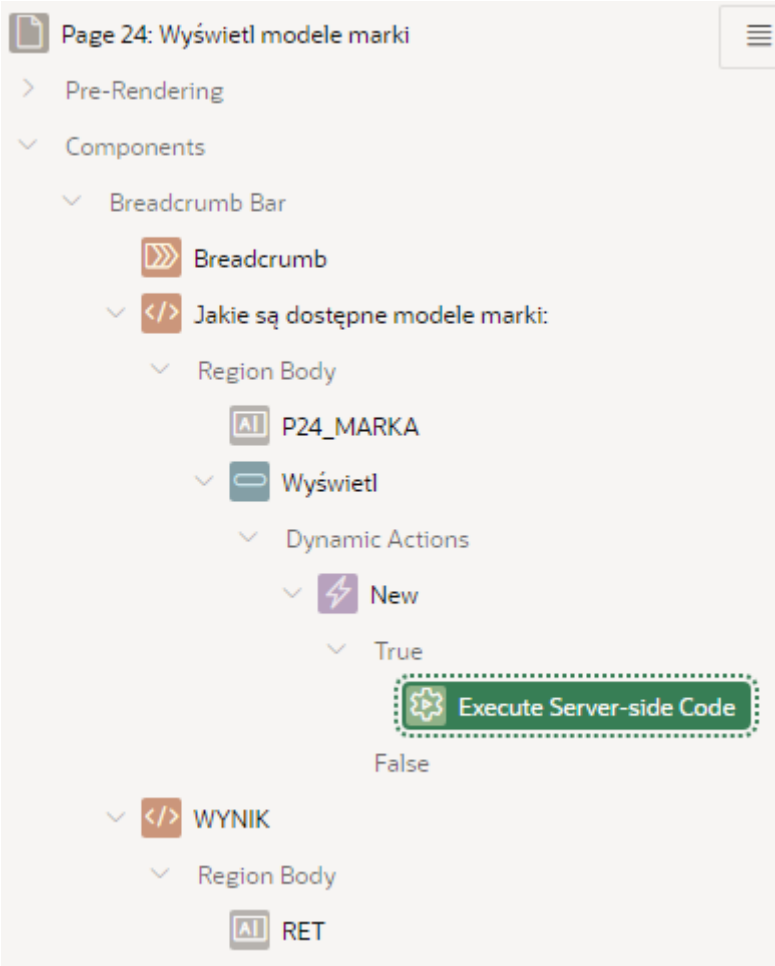
Wygląd:

Średni rocznik samochodów danej marki
Średni rocznik
Marka
Wyświetl
WYNIK
Ajax call returned server error ORA-20005: Nie podano marki samochodu for Execute Server-Side Code.
OK

Średni rocznik samochodów danej marki
Średni rocznik
Marka
BMWe
Wyświetl
WYNIK
Ajax call returned server error ORA-20002: Brak samochodów marki BMWe w bazie danych for Execute Server-Side Code.
OK

24. Wyświetl modele marki

Page Designer:



Kod:

```
BEGIN
   :RET :=
pakiet3.wyswietl_modele_marki(:P24_MARKA
);
END;
```

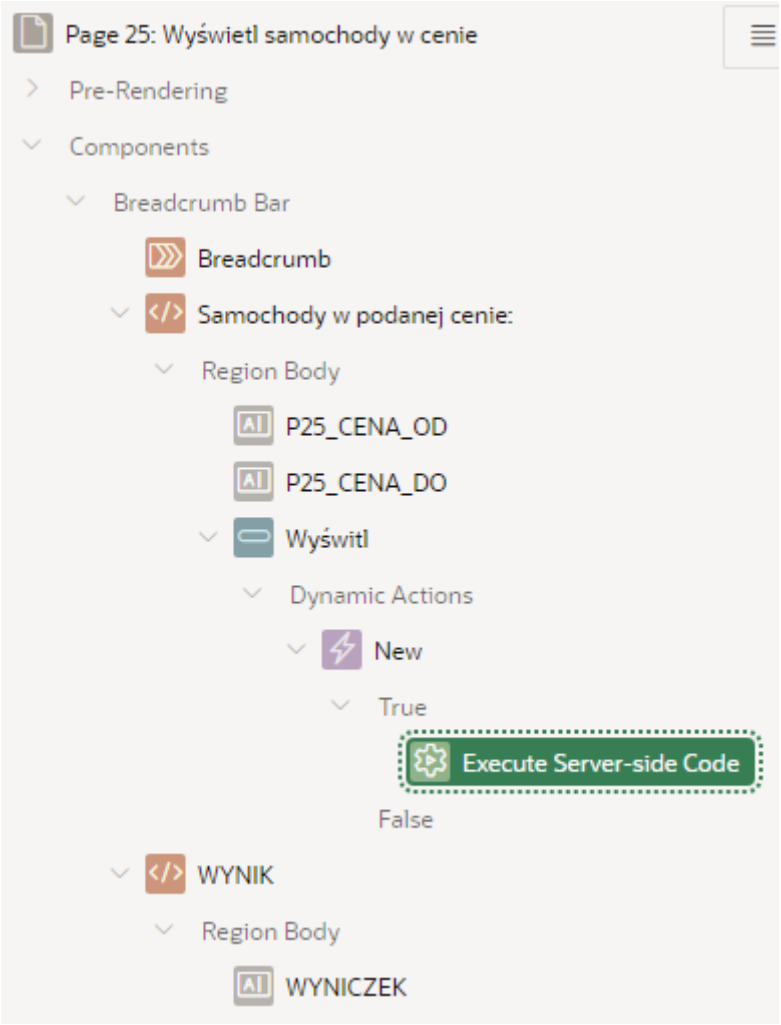
Wygląd:

Wyświetl modele marki
Jakie są dostępne modele marki:
Marka
Wyświetl
WYNIK
Ajax call returned server error ORA-20005: Podaj markę samochodu! for Execute Server-Side Code.
OK

Wyświetl modele marki
Jakie są dostępne modele marki:
Marka
Poloneze
Wyświetl
WYNIK
Ajax call returned server error ORA-20003: Nie znaleziono Poloneze w naszym komisie. for Execute Server-Side Code.
OK

25. Wyświetl samochody w cenie

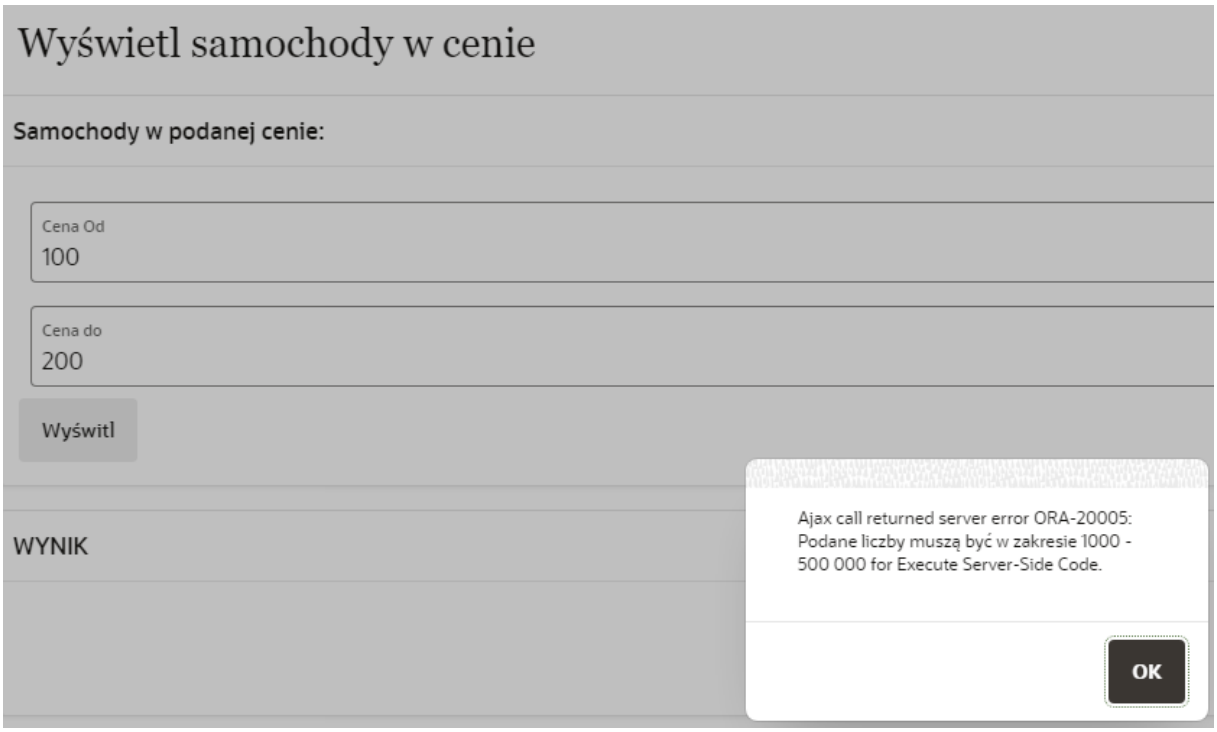
Page Designer:

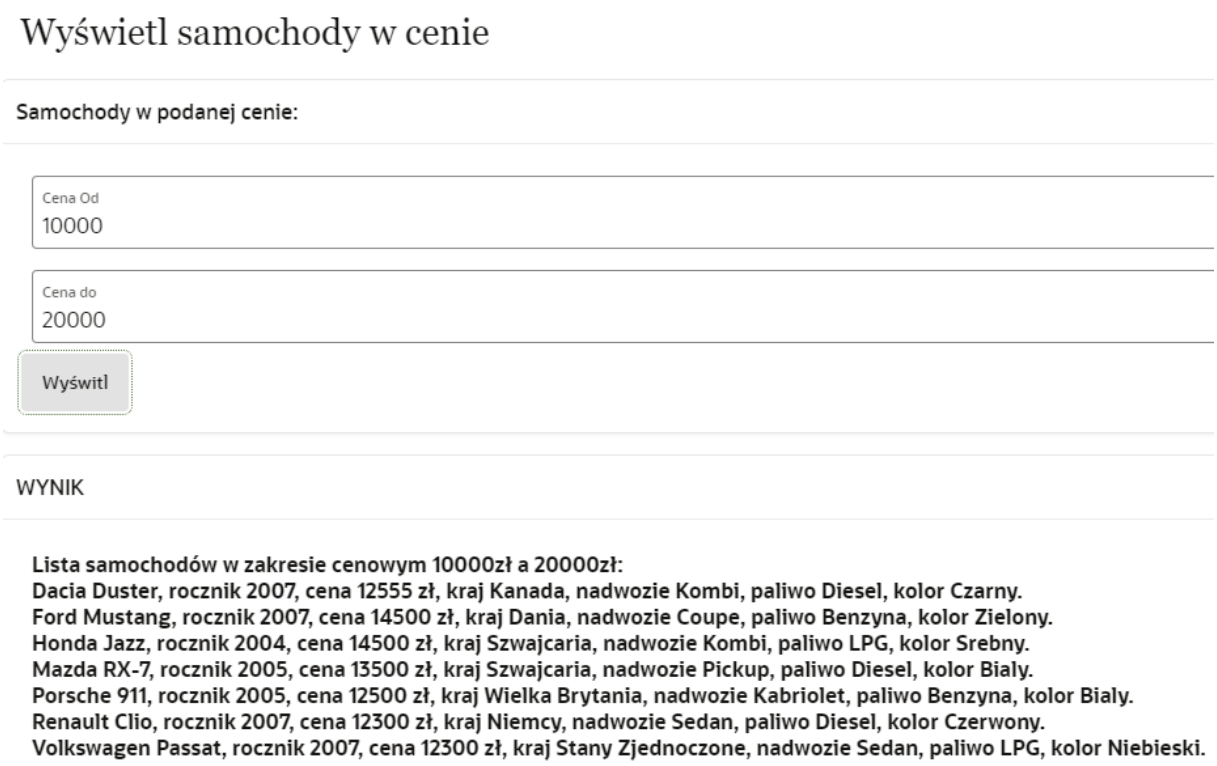


Kod:

```
BEGIN
   :WYNICZEK :=
pakiet3.wyswietl_samochody_w_cenie(:P25_CENA_OD,
 :P25_CENA_DO
);
END;
```

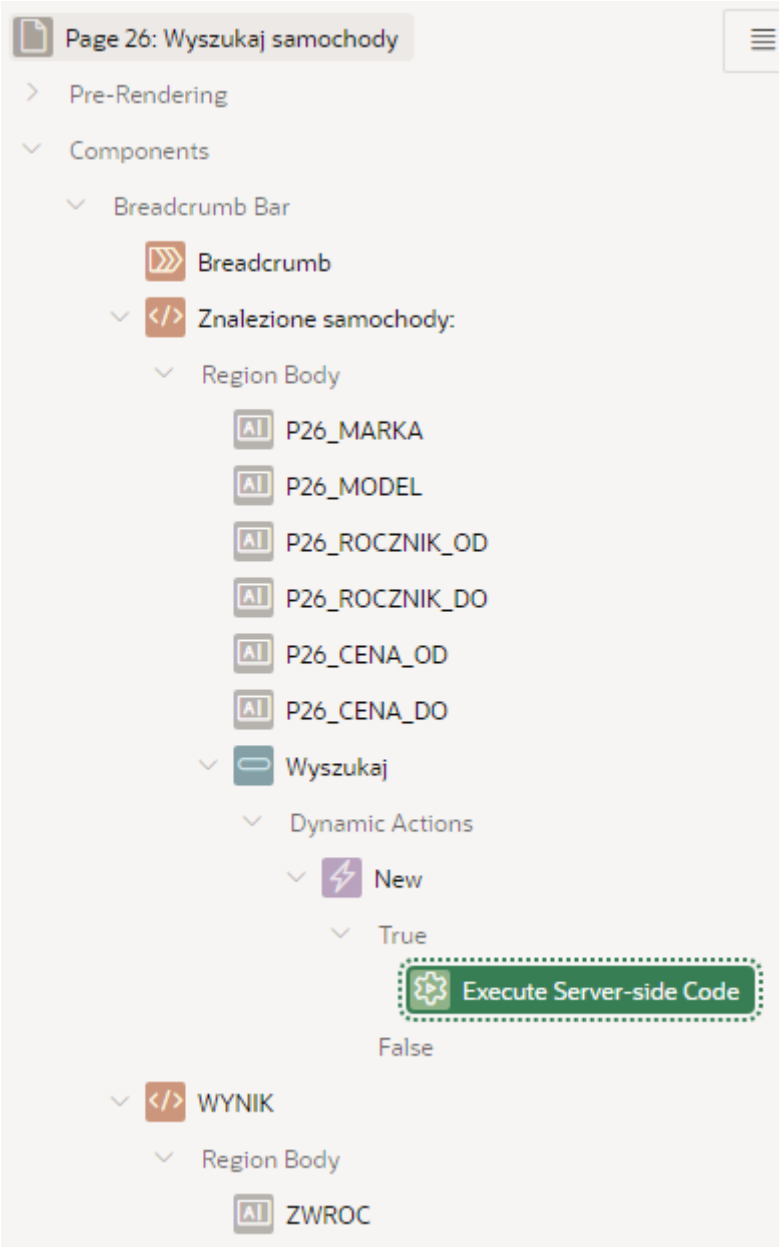
Wygląd:





26. Wyszukaj samochody

Page Designer:


Page 26: Wyszukaj samochody
Pre-Rendering
Components
Breadcrumb Bar
Breadcrumb
Znalezione samochody:
Region Body
P26_MARKA
P26_MODEL
P26_ROCZNIK_OD
P26_ROCZNIK_DO
P26_CENA_OD
P26_CENA_DO
Wyszukaj
Dynamic Actions
New
True
Execute Server-side Code
False
WYNIK
Region Body
ZWROC

Kod:

```
BEGIN
    :ZWROC :=
pakiet3.wyszukaj_samochody(
     :P26_MARKA,
     :P26_MODEL,
     :P26_ROCZNIK_OD,
     :P26_ROCZNIK_DO,
     :P26_CENA_OD,
     :P26_CENA_DO
);
END;
```

Wygląd:

Wyszukaj samochody

Znalezione samochody:

MARKA
Audi

MODEL

ROCZNIK_OD

ROCZNIK_DO

CENA_OD

CENA_DO

Wyszukaj

Ajax call returned server error ORA-20004: Nie znaleziono modelu: w naszym komisie. for Execute Server-Side Code.

OK

Wyszukaj samochody

Znalezione samochody:

MARKA
Audi

MODEL
A4

ROCZNIK_OD
1870

ROCZNIK_DO
2023

CENA_OD

CENA_DO

Wyszukaj

Ajax call returned server error ORA-20004: Podane lata są błędne. for Execute Server-Side Code.

OK

Wyszukaj samochody

Znalezione samochody:

MARKA
Audi

MODEL
A4

ROCZNIK_OD
1970

ROCZNIK_DO
2023

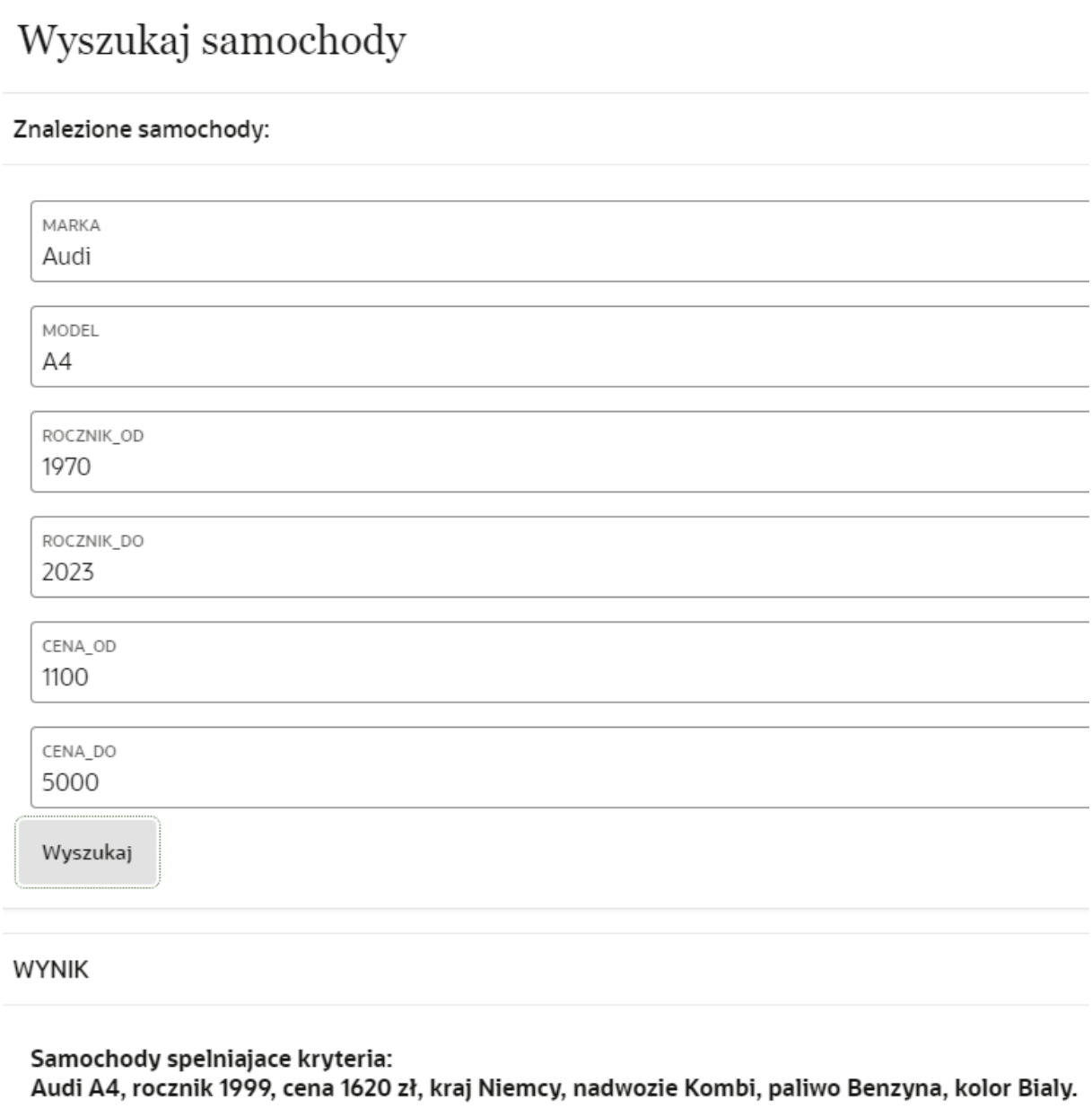
CENA_OD
100

CENA_DO
200

Wyszukaj

Ajax call returned server error ORA-20004: Cena musi byc w zakresie 1000 - 500 000. for Execute Server-Side Code.

OK



7. Podsumowanie

Po zakończeniu projektu powstała nowoczesna aplikacja komis samochodowy na platformie Oracle APEX, która pozwala na zarządzanie ofertą samochodów, obsługę transakcji oraz generowanie raportów i analizowanie danych sprzedaży. Aplikacja ta jest przyjazna dla użytkownika i łatwa w obsłudze, a jej funkcjonalności pozwalają na usprawnienie procesów związanych z działalnością komisu samochodowego. Wszystkie funkcje i procedury zostały przetestowane i działają poprawnie, co pozwala na swobodne korzystanie z aplikacji przez użytkowników. Dzięki tej aplikacji, komis samochodowy jest w stanie skuteczniej prezentować swoją ofertę, obsługiwać transakcje oraz analizować dane sprzedaży, co przyczynia się do lepszego zarządzania biznesem.

Należy pamiętać, że platforma Oracle APEX pozwala na zrobienie dużo więcej niż tylko aplikację komis samochodowy, jednak w ramach tego projektu skupiliśmy się na konkretnych potrzebach i ograniczyliśmy się do wybranych funkcjonalności. Warto pamiętać, że jest to dopiero początek i istnieje dużo możliwości rozbudowy tej aplikacji, jednak ze względu na ograniczony czas i to, że dopiero uczyliśmy się tej platformy, nie byliśmy w stanie wykorzystać wszystkich możliwości.