Temat: System zarządzania komisem samochodowym

Autorzy: Szymon Budziosz, Piotr Knapik

Projekt bazy danych dla komisu samochodowego Bazy danych Prowadzący: dr inż. Robert Marcjan Autorzy: Szymon Budziosz, Piotr Knapik Grupa 1K333/SI Informatyka (Niestacjonarne) Rok III.
Kod bazy : https://github.com/kosmo8019/sql-komis

1.Opis systemu

1. Aplikacja jest przeznaczona do ewidencjonowania pojazdów, klientów oraz rejestrowania transakcji zakupu i sprzedaży pojazdów. Umożliwia prowadzenie kartoteki klientów komisu, a także ewidencjonuje pojazdy, które przeszły przez komis. Dostarcza danych do wystawienia faktury kupna/sprzedaży, a także umożliwia określenie, które pojazdy są na sprzedaż. Pozwala w łatwy sposób znaleźć oferty odpowiadające oczekiwaniom klienta poprzez zastosowane filtry. Klient z łatwością może je porównać między sobą. Kontakt pomiędzy dealerem, a kupującym dzięki aplikacji będzie szybszy i bardziej rzeczowy . Klient dowie się o możliwości kredytowania zakupu samochodu. Pozna możliwości ubezpieczenia (dane ubezpieczyciela, koszt, zakres ubezpieczenia).
2. Przepływ danych Klient Zamówienie/ oferta sprzedaży Przyjęcie zgłoszenia kupna/sprzedaży Dane o zgłoszeniu Weryfikacja zgłoszenia: Dobór odpowiadającego klientowi auta (przedstawienie oferty sprzedaży)/ Sprawdzenie stanu technicznego i wycena w celu zakupu Przegląd dostępnych aut Finalizowanie transakcji Kartoteka klientów Sprzedaż / zakup auta Kartoteka pojazdów Dane o kliencie Dane o pojeździe Dane o transakcji Kartoteka transakcji kupna/sprzedaży.
Definicja bytów z określeniem ich cech. Klient (nazwisko, imię, pesel, , kod pocztowy, miasto, ulica, nr domu, nr lokalu, adres e-mail, numer telefonu). Pojazd (marka, model, rok produkcji, nr Vin , przebieg, pojemność, rodzaj paliwa, kolor, nadwozie, wyposażenie). Transakcja (rodzaj transakcji, data transakcji, kwota brutto, , id faktury, sposób zapłaty, czy zapłacono).

2. Przypadki użycia

1. Diagram przypadków użycia

2. Opis przypadków użycia

Nazwa: dodanie użytkownika klient Aktorzy: klient ,sprzedawca
Scenariusz: 1.Aktor wpisuje na stronie rejestracji swoje imię , nazwisko , adres email i tel. 2.Klika na przycisk ‘rejestracja’ 3.System sprawdza unikalność danych 4.System generuje dla użytkownika hasło oraz login i wyświetla w oknie przeglądarki

Nazwa: usuwanie użytkownika z systemu Aktorzy: klient ,sprzedawca
Scenariusz: 1.Aktor loguje się do systemu 2.Aktor klika na zakładkę dane użytkownika i przycisk usuń dane z systemu

Nazwa: dodanie użytkownika sprzedawca Aktorzy: administrator
Scenariusz: 1.Aktor loguje się do systemu 2.Aktor wyszukuje użytkownika 3.Aktor nadaje uprawnienia z zakresu – administrator ,sprzedawca

Nazwa: utworzenie profilu użytkownika Aktorzy: klient ,sprzedawca
Scenariusz: 1.Aktor loguje się do systemu 2.W zakładce ustawienia użytkownika Aktor określa swoje ogólne preferencje co do samochodu takie jak - model,rok_produkcji,przebieg_do,pojemnosc_od,pojemnosc_do,paliwo,nadwozie oraz maksymalnie 10 dodatków szczegółowych: `( '2 drzwi'),('3 drzwi'),('4/5 drzwi'),('2 miejsca'),('5 miejsc'),('7 miejsc'),('na koła przednie'),('na koła tylne'),('4x4'),('manualna'),('automatyczna'),('klimatyzacja`

manualna'),('brak klimatyzacji'),('klimatyzacja automatyczna'),('oświetlenie do jazdy dziennej'),('światła led'),('światła przeciwmgłowe'),('światła ksenonowe'),('ABS'),('ESP'),('kontrola trakcji'),('asystent parkowania'),('czujniki parkowania'),('czujnik deszczu'),('immobilizer'),('poduszka pasażera'),('poduszka boczna'),(‘kamera cofania'),('elektryczne szyby przednie'),('elektryczne szyby tylne'),('elektryczne lusterka'),('podgrzewane lusterka'),('kierownica wielofunkcyjna'),('podgrzewane siedzenia'),('wspomaganie kierownicy'),('przyciemniane szyby'),('radio fabryczne'),('radio niefabryczne'),('mp3'),('zmieniarka CD'),('nawigacja GPS'),('bluetooth'),('alufelgi'),('centralny zamek'),('panoramiczny dach'),('hak'),('komputer pokładowy'),('tempomat'),('metalik'),('akryl'),('matowy'),('perłowy'),('krajowe'),('importowany'),('bezwypadkowy'),('pierwszy właściciel'),('serwisowany w aso'),('zarejestrowany w polsce'),('czarny'),('czerwony'),('zielony'),('biały'),('srebrny'),('niebieski')
3.Klika przycisk ‘Zapisz’, system wysyła dane do bazy danych

Nazwa: modyfikacja profilu użytkownika
Aktorzy: klient

Scenariusz:
1.Aktor loguje się do systemu
2.W zakładce ustawienia użytkownika Aktor zmienia swoje preferencje co do samochodu
3.Zapisuje, system wysyła dane do bazy danych

Nazwa: dodanie samochodu do bazy
Aktorzy: sprzedawca

Scenariusz:
1.W zakładce DODAJ Aktor wpisuje dane dotyczące samochodu który ma znaleźć się w komisie:
([producent],[model],[nadwozia],[cena],[przebieg],[data_produkcji],[kraj_pochodzenia],[ksiazka_serwisowa],[pojemnosc],[paliwo])
3.Zapisuje, system wysyła dane do bazy danych

Nazwa: usunięcie samochodu z bazy
Aktorzy: sprzedawca

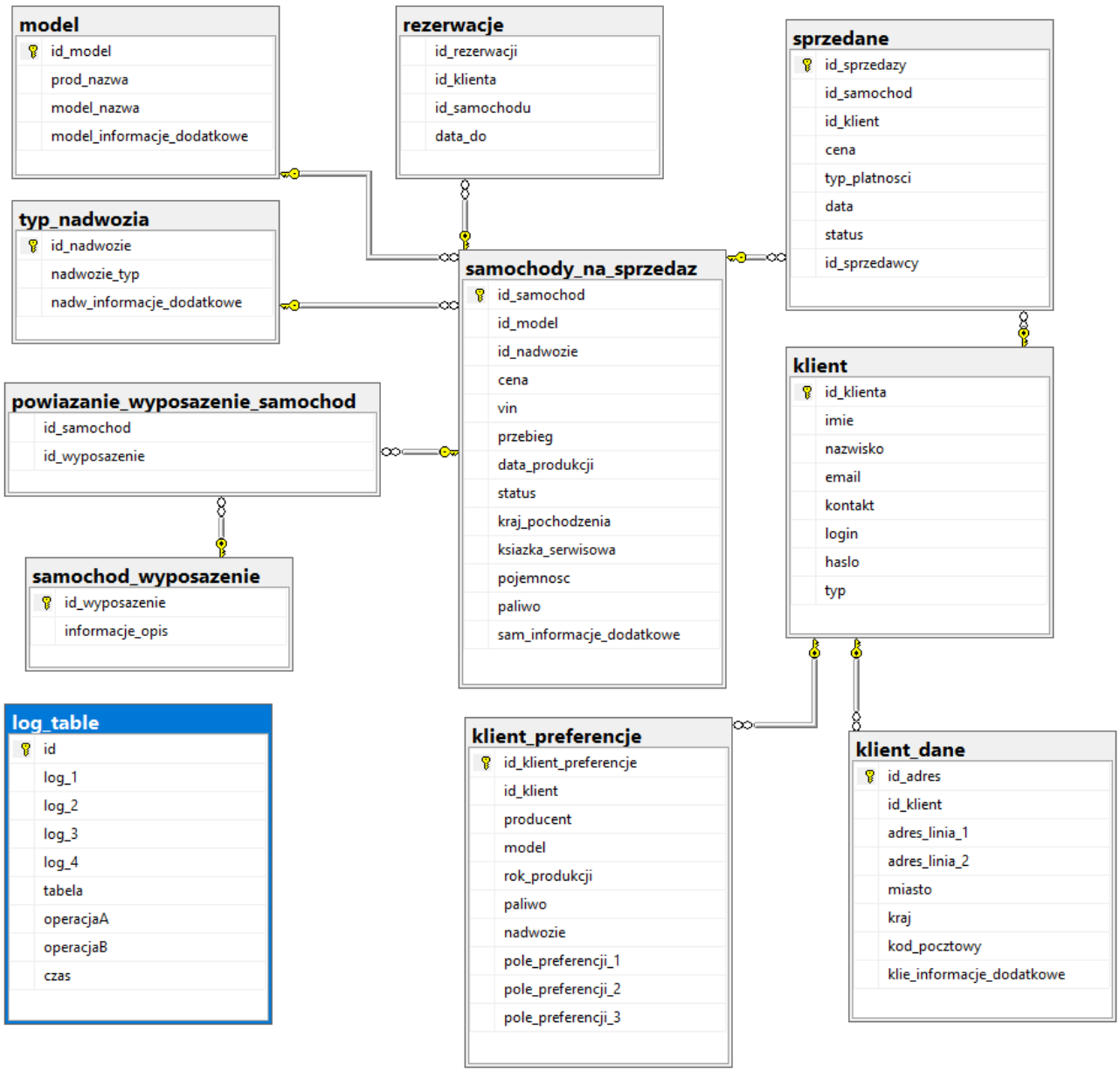
Scenariusz:
1.Aktory wyszukuje samochod po nr. Id lub vin
2.Zapisuje, system wysyła dane do bazy danych

Nazwa: sprzedaż samochodu Aktorzy: sprzedawca
Scenariusz: 2.W zakładce SPRZEDAŻ Aktor wpisuje dane do wyszukania samochodu (nr.rejestracyjny ,właściciel ,nr.umowy) oraz dane kupującego (nazwisko,pesel) 3.Wygenerowanie umowy na podstawie wyszukanych danych 4.Zapisuje, system wysyła dane do bazy danych

Nazwa: Logowanie Aktorzy: klient ,sprzedawca ,administrator
Scenariusz: 1. Aktor wpisuje login i hasło w formularzu logowania 2. Aktor klika przycisk ‘login’ 3. Aktor otrzymuje dostęp do systemu wg. Uprawnień przypisanych do jego loginu

Nazwa: płatność Aktorzy: sprzedawca
1.Aktor w zakładce ‘płatności’ wybiera formę płatności (gotówka , kredyt) 2.Aktor wpisuje kwotę należności i klika przycisk akceptuj

SCHEMAT BAZY DANYCH



Tabele

[dbo].[klient]

KOLUMNA	TYP	OPIS
[id_klienta]	[int]	KLUCZ GŁÓWNY
[imie]	[varchar](30)	
[nazwisko]	[varchar](50)	
[email]	[varchar](50)	
[kontakt]	[varchar](20)	
[login]	[varchar](15)	GENEROWANE PRZEZ PROCEDURĘ
[haslo]	[varchar](15)	GENEROWANE PRZEZ PROCEDURĘ
[typ]	[char](1)	D-DEALER S-KLIENT A- ADMIN

[dbo].[klient_dane]

KOLUMNA	TYP	OPIS
[id_adres]	[int]	KLUCZ GŁÓWNY
[id_klient]	[int]	KLUCZ OBCY
[adres_linia_1]	[varchar](50)	
[adres_linia_2]	[varchar](50)	
[miasto]	[varchar](40)	
[kraj]	[varchar](25)	
[kod_pocztowy]	[varchar](15)	
[klie_informacje_dodatkowe]	[varchar](50)	

[dbo].[klient_preferencje]

KOLUMNA	TYP	OPIS
[id_klient_preferencje]	[int]	KLUCZ GŁÓWNY
[id_klient]	[int]	KLUCZ OBCY
[producent]	[varchar](50)	W POLA KLIENT MOŻE WPISAĆ ATRYBUTY WEDŁUG KTÓRYCH MOŻLIWE JEST WYCZUKANIE SAMOCHODU POLA PREFERENCJI PRZYJMUJĄ WARTOŚC LICZBOWĄ ODPPOWIADAJACĄ TABELI [samochod_wyposazenie]
[model]	[varchar](50)	
[rok_produkcji]	[int]	
[paliwo]	[char](1)	
[nadwozie]	[int]	
[pole_preferencji_1]	[int]	
[pole_preferencji_2]	[int]	
[pole_preferencji_3]	[int]	

[dbo].[samochody_na_sprzedaz]

KOLUMNA	TYP	OPIS
[id_samochod]	[int]	KLUCZ GŁÓWNY
[id_model]	[int]	KLUCZ OBCY
[id_nadwozie]	[int]	KLUCZ OBCY
[cena]	[int]	CENA ŻĄDANA
[vin]	[varchar](30)	
[przebieg]	[int]	
[data_produkcji]	[int]	ROK PRODUKCJI
[status]	[char](1)	S-SPRZEDANY,R-REZERWACJA,W-WOLNY
[kraj_pochodzenia]	[char](3)	
[ksiazka_serwisowa]	[char](1)	T-TAK,N-NIE
[pojemnosc]	[int]	
[paliwo]	[char](1)	B,G,O
[sam_informacje_dodatkowe]	[varchar](50)	

[dbo].[rezerwacje]

KOLUMNA	TYP	OPIS
[id_rezerwacji]	[int]	KLUCZ GŁÓWNY
[id_klienta]	[int]	
[id_samochodu]	[int]	
[data_do]	[smalldatetime]	DATA KOŃCA REZERWACJI

[dbo].[sprzedane]

KOLUMNA	TYP	OPIS
[id_sprzedazy]	[int]	KLUCZ GŁÓWNY
[id_samochod]	[int]	KLUCZ OBCY
[id_klient]	[int]	KLUCZ OBCY
[cena]	[int]	KWOTA OTRZYMANA
[typ_platnosci]	[char](1)	L-LEASING,G-GOTÓWKA,K-KREDYT
[data]	[date]	DATA SPRZEDAŻY
[status]	[char](1)	T,N
[id_sprzedawcy]	[int]	ID DEALERA DOKONUJACEGO SPRZEDAZY

[dbo].[log_table]

KOLUMNA	TYP	OPIS
[id]	[int]	KLUCZ GŁÓWNY
[log_1]	[varchar](50)	POLA NA INFORMACJE LOGOWANE
[log_2]	[varchar](50)	
[log_3]	[varchar](50)	
[log_4]	[varchar](50)	
[tabela]	[varchar](50)	NAZWA TABELI NA KTÓREJ WYKONANO OPERACJE
[operacjaA]	[varchar](20)	TYP OPERACJI:INSERT/DELETE /UPDATE LUB NAZWA UŻYTKOWNIKA
[operacjaB]	[varchar](20)	
[czas]	[smalldatetime]	DATA OPERACJI

[dbo].[samochod_wyposazenie]

KOLUMNA	TYP	OPIS
[id_wyposazenie]	[int]	KLUCZ GŁÓWNY
[informacje_opis]	[varchar](50)	NAZWA WYPOSAŻENIA

[dbo].[typ_nadwozia]

KOLUMNA	TYP	OPIS
[id_nadwozie]	[int]	KLUCZ GŁÓWNY
[nadwozie_typ]	[char](20)	NAZWA NADWOZIA
[nadw_informacje_dodatkowe]	[varchar](50)	

[dbo].[model]

KOLUMNA	TYP	OPIS
[id_model]	[int]	KLUCZ GŁÓWNY
[prod_nazwa]	[varchar](50)	NAZWA PRODUCENTA
[model_nazwa]	[char](50)	NAZWA MODELU
[model_informacje_dodatkowe]	[varchar](200)	

[dbo].[powiazanie_wyposazenie_samochod]

KOLUMNA	TYP	OPIS
[id_samochod]	[int]	KLUCZ OBCY
[id_wyposazenie]	[int]	KLUCZ OBCY

WIDOKI

a. Widok przedstawia podstawowe dane klientów (S)

```
CREATE VIEW [dbo].[v_klient_dane]
AS
     SELECT dbo.klient.imie,
            dbo.klient.nazwisko,
            dbo.klient.email,
            dbo.klient.kontakt,
            dbo.klient_dane.adres_linia_1,
            dbo.klient_dane.adres_linia_2,
            dbo.klient_dane.miasto,
            dbo.klient_dane.kod_pocztowy,
            dbo.klient.typ,
            dbo.klient.id_klienta
     FROM dbo.klient
          LEFT OUTER JOIN dbo.klient_dane ON
             dbo.klient.id_klienta = dbo.klient_dane.id_klient
     WHERE(dbo.klient.typ = 'S')
```

	imie	nazwisko	email	kontakt	adres_linia_1	adres_linia_2	miasto	kod_pocztowy	typ	id_klienta
1	Dalis	Boss	dboss6@utexas.edu	9338599513	85 Merchant Lane	NULL	Campamento	NULL	S	7
2	Comall	Carlon	ccarlon7@ox.ac.uk	5419016710	NULL	NULL	NULL	NULL	S	8
3	Grier	Gascoyne	ggascoyne8@linkedin.com	6738059972	5179 Victoria Drive	NULL	Fushun	NULL	S	9

b. Widok preferencji klientów

```
CREATE VIEW dbo.v_klienci_preferencje
AS
     SELECT dbo.klient.imie,
            dbo.klient.nazwisko,
            dbo.klient_preferencje.producent,
            dbo.klient_preferencje.model,
            dbo.klient_preferencje.rok_produkcji,
            dbo.klient_preferencje.paliwo,
(
    SELECT nadwozie_typ
    FROM dbo.typ_nadwozia
    WHERE dbo.klient_preferencje.nadwozie = id_nadwozie
) AS Expr1,
(
    SELECT informacje_opis
    FROM dbo.samochod_wyposazenie
    WHERE dbo.klient_preferencje.pole_preferencji_1 = id_wyposazenie
) AS Pref_1,
(
    SELECT informacje_opis
    FROM dbo.samochod_wyposazenie AS samochod_wyposazenie_2
```

```
    WHERE dbo.klient_preferencje.pole_preferencji_2 = id_wyposazenie
) AS Pref_2,
(
    SELECT informacje_opis
    FROM dbo.samochod_wyposazenie AS samochod_wyposazenie_1
    WHERE dbo.klient_preferencje.pole_preferencji_3 = id_wyposazenie
) AS Pref_3
     FROM dbo.klient
          INNER JOIN dbo.klient_preferencje ON dbo.klient.id_klienta =
dbo.klient_preferencje.id_klient
```

	imie	nazwisko	producent	model	rok_produkcji	paliwo	Expr1	Pref_1	Pref_2	Pref_3
1	Doe	Stiegars	Mitsubishi	Challenger	2003	O	limuzyna	2 drzwi	ABS	radio fabryczne
2	Wilton	Bridewell	Dodge	D350 Club	1993	O	suv	3 drzwi	NULL	radio niefabryczne
3	Julieta	Cud	Mitsubishi	Chariot	1992	B	sedan	4/5 drzwi	NULL	mp3
4	Karrie	Profit	Hummer	H2	2009	G	kombi	2 miejsca	NULL	zmieniarka CD
5	Stillmann	Schiell	Toyota	Solara	2006	B	sedan	5 miejsc	NULL	nawigacja GPS

c. Widok najpopularniejszych marek wybranych przez klientów w swoich preferencjach

```
CREATE VIEW [dbo].[v_preferencje_popularni_producenci]
AS
     SELECT TOP 10 COUNT(producent) AS sztuk,
                   producent
     FROM v_klienci_preferencje
     GROUP BY producent
            ORDER BY sztuk DESC
```

	sztuk	producent
1	7	Ford
2	4	Dodge
3	4	Mitsubishi
4	3	Toyota
5	2	Buick
6	2	BMW
7	2	Chevrolet
8	2	Hyundai
9	2	Hummer
10	2	Lexus

d. Widok pokazuje ilość samochodów poszczególnych marek na stanie komisu

```
CREATE VIEW [dbo].[v_marki_na_stanie]
AS
     SELECT prod_nazwa AS marka,
            COUNT(*) AS ilość_na_stanie
     FROM dbo.v_samochody_do_sprzedania
     GROUP BY prod_nazwa,
              status;
GO
```

	marka	ilość_na_stanie
1	Acura	10
2	Aston Martin	1
3	Audi	3
4	Bentley	1
5	BMW	2
6	Buick	2
7	Cadillac	4
8	Chevrolet	17
9	Chrysler	3
10	Dodge	10

e. Widok samochodów do sprzedania (bez rezerwacji)

```
CREATE VIEW [dbo].[v_samochody_do_sprzedania]
AS
     SELECT dbo.model.prod_nazwa,
            dbo.model.model_nazwa,
            dbo.samochody_na_sprzedaz.cena,
            dbo.samochody_na_sprzedaz.przebieg,
            dbo.samochody_na_sprzedaz.data_produkcji,
            dbo.samochody_na_sprzedaz.ksiazka_serwisowa,
            dbo.samochody_na_sprzedaz.pojemnosc,
            dbo.samochody_na_sprzedaz.paliwo,
            dbo.typ_nadwozia.nadwozie_typ,
            dbo.samochody_na_sprzedaz.status,
            dbo.samochody_na_sprzedaz.id_samochod
     FROM dbo.model
          LEFT OUTER JOIN dbo.samochody_na_sprzedaz ON dbo.model.id_model =
                          dbo.samochody_na_sprzedaz.id_model
          LEFT OUTER JOIN dbo.typ_nadwozia ON dbo.samochody_na_sprzedaz.id_nadwozie =
                          dbo.typ_nadwozia.id_nadwozie
     WHERE(dbo.samochody_na_sprzedaz.status = 'W');
GO
```

	prod_nazwa	model_nazwa	cena	przebieg	data_produkcji	ksiazka_serwisowa	pojemnosc	paliwo	nadwozie_typ	status	id_samochod
1	Acura	TL	76869	200000	1986	T	3000	G	kombi	W	344
2	Acura	TL	54678	12000	1995	T	1000	B	sedan	W	411
3	Mazda	CX-7	30196	60000	1989	N	1600	G	sedan	W	328
4	Subaru	Loyale	96786	30000	1984	N	1300	G	kombi	W	428
5	Lotus	Esprit Turbo	93422	300000	1986	N	4000	G	sedan	W	282
6	Suzuki	SJ	46655	130000	2012	T	1800	B	suv	W	414

f. Widok przedstawia wszystkie sprzedane samochody

```
CREATE VIEW [dbo].[v_samochody_sprzedane]
AS
     SELECT dbo.model.prod_nazwa,
            dbo.model.model_nazwa,
            dbo.samochody_na_sprzedaz.id_model,
            dbo.samochody_na_sprzedaz.cena,
            dbo.samochody_na_sprzedaz.przebieg,
            dbo.samochody_na_sprzedaz.data_produkcji,
            dbo.samochody_na_sprzedaz.ksiazka_serwisowa,
            dbo.samochody_na_sprzedaz.pojemnosc,
            dbo.samochody_na_sprzedaz.paliwo,
            dbo.typ_nadwozia.nadwozie_typ,
```

```
        dbo.samochody_na_sprzedaz.id_samochod
FROM dbo.model
     INNER JOIN dbo.samochody_na_sprzedaz ON dbo.model.id_model =
                dbo.samochody_na_sprzedaz.id_model
     INNER JOIN dbo.typ_nadwozia ON dbo.samochody_na_sprzedaz.id_nadwozie =
                dbo.typ_nadwozia.id_nadwozie
WHERE(dbo.samochody_na_sprzedaz.status = 'S')
```

	prod_nazwa	model_nazwa	id_model	cena	przebieg	data_produkcji	ksiazka_serwisowa	pojemnosc	paliwo	nadwozie_typ	id_samochod
1	Ford	Explorer	237	117652	30000	1989	N	1600	B	sedan	223
2	GMC	Savana 1500	185	30084	300000	2012	T	1800	B	sedan	227
3	Volvo	XC70	160	141668	30000	2004	T	1000	O	sedan	231
4	Mercedes-Benz	190E	175	62445	200000	1998	T	1600	O	kombi	235
5	Hyundai	Elantra	253	96268	200000	1998	N	1600	O	suv	239

g. Widok przedstawia raport pogrupowany po typach płatności (kredyt,leasing,gotówka)

```
CREATE VIEW [dbo].[v_raport_typu_platnosci]
AS
     SELECT SUM(cena) AS Suma,
            typ_platnosci,
            COUNT(prod_nazwa) AS ilość_sprzedanych_samochodów
     FROM dbo.v_raport_sprzedaz
     GROUP BY typ_platnosci
```

	Suma	typ_platnosci	ilość_sprzedanych_samochodów
1	610000	G	12
2	1145000	K	18
3	1030000	L	14
4	99000	t	1

h. Raport sprzedaży szczegółowy

```
CREATE VIEW [dbo].[v_raport_sprzedaz]
AS
     SELECT dbo.model.prod_nazwa,
            dbo.model.model_nazwa,
            dbo.samochody_na_sprzedaz.vin,
            dbo.sprzedane.cena,
            dbo.sprzedane.data,
            dbo.sprzedane.status,
            dbo.sprzedane.id_sprzedawcy,
            dbo.sprzedane.typ_platnosci,
            dbo.klient.imie,
            dbo.klient.nazwisko
     FROM dbo.samochody_na_sprzedaz
          INNER JOIN dbo.sprzedane ON dbo.samochody_na_sprzedaz.id_samochod =
                     dbo.sprzedane.id_samochod
          INNER JOIN dbo.model ON dbo.samochody_na_sprzedaz.id_model = dbo.model.id_model
          INNER JOIN dbo.klient ON dbo.sprzedane.id_sprzedawcy = dbo.klient.id_klienta
```

	prod_nazwa	model_nazwa	vin	cena	data	status	id_sprzedawcy	typ_platnosci	imie	nazwisko
1	Ford	Explorer	1C4RJEAG2EC129439	50000	2018-04-02	T	5	L	Barbara-anne	Bushrod
2	GMC	Savana 1500	WBASP4C53BC331883	100000	2017-10-20	T	5	L	Barbara-anne	Bushrod
3	Volvo	XC70	WA1TFAFL4EA748720	45000	2017-05-10	T	1	L	Drew	Sidwick
4	Mercedes-Benz	190E	YV1612FH2D2307643	15000	2017-05-28	T	5	L	Barbara-anne	Bushrod
5	Hyundai	Elantra	WBALM73569E994331	140000	2017-10-07	T	1	L	Drew	Sidwick
6	Mitsubishi	Galant	1FTEX1CM5DK826713	80000	2017-11-02	T	3	L	Blakeley	Chinery
7	Kia	Sephia	WDDGF5EB7AR165387	50000	2017-07-03	T	1	G	Drew	Sidwick
8	Chevrolet	Beretta	1FTFW1E86AK104556	50000	2017-10-21	T	4	L	Doralyn	Jackson
9	Honda	S2000	WBADX7C5XCE184283	50000	2018-04-04	T	4	L	Doralyn	Jackson

i. Raport sprzedaży ogólny

```
CREATE VIEW [dbo].[wyniki_sprzedawcy]
AS
     SELECT imie+' '+nazwisko AS Sprzedawca,
            SUM(cena) AS suma_sprzedazy
     FROM dbo.v_raport_sprzedaz
     GROUP BY nazwisko,
              imie
```

	Sprzedawca	suma_sprzedazy
1	Barbara-anne Bushrod	605000
2	Blakeley Chinery	300000
3	Doralyn Jackson	705000
4	Drew Sidwick	524000
5	Esteban Abraham	155000
6	Thane Wince	595000

j. Raport rezerwacji

```
CREATE VIEW [dbo].[v_rezerwacje_klient_samochod]
AS
     SELECT dbo.klient.imie,
            dbo.klient.nazwisko,
            dbo.klient.kontakt,
            dbo.rezerwacje.id_samochodu,
            dbo.rezerwacje.id_klienta,
            dbo.rezerwacje.data_do,
            dbo.samochody_na_sprzedaz.vin,
            dbo.samochody_na_sprzedaz.status,
            dbo.model.prod_nazwa,
            dbo.model.model_nazwa
     FROM dbo.klient
          INNER JOIN dbo.rezerwacje ON dbo.klient.id_klienta = dbo.rezerwacje.id_klienta
          INNER JOIN dbo.samochody_na_sprzedaz ON dbo.rezerwacje.id_samochodu =
                     dbo.samochody_na_sprzedaz.id_samochod
          INNER JOIN dbo.model ON dbo.samochody_na_sprzedaz.id_model = dbo.model.id_model
```

	imie	nazwisko	kontakt	id_samochodu	id_klienta	data_do	vin	status	prod_nazwa	model_nazwa
1	Mommy	Steel	7541776674	417	199	2018-06-09 00:00:00	19UYA42712A960170	R	GMC	2500
2	Dario	Lumm	1224525180	418	198	2018-06-05 00:00:00	3N1CN7AP6EK903193	R	Volkswagen	riolet

PROCEDURY

a.procedura dodania klienta , procedura zwraca haslo i login dla kienta

```
CREATE PROC dbo.dodaj_klienta @imie    VARCHAR(15),
                              @nazw    VARCHAR(20),
                              @email   VARCHAR(20),
                              @kontakt VARCHAR(20),
                              @typ     CHAR(1)
AS
     BEGIN
         DECLARE @output VARCHAR(10), @login VARCHAR(10), @haslo VARCHAR(10);
         EXEC gen_haslo_login
              @output OUTPUT;
         SET @login = @output;
         EXEC gen_haslo_login
              @output OUTPUT;
         IF @imie IS NULL OR @nazw IS NULL OR @kontakt IS NULL
             BEGIN
                 PRINT 'BRAK ARGUMENTU';
                 RETURN;
             END;
         IF EXISTS
(
    SELECT * FROM klient
    WHERE imie = @imie AND nazwisko = @nazw
)
             BEGIN
                 PRINT 'uzytkownik'+@imie+@nazw+'jest już w systemie';
                 RETURN;
             END;
         IF EXISTS
(
    SELECT * FROM klient
    WHERE email = @email
)
             BEGIN
                 PRINT 'adres mejlowy '+@email+'jest już w systemie';
                 RETURN;
             END;
         INSERT INTO klient (imie, nazwisko, email, kontakt, login, haslo, typ)
         VALUES (@imie, @nazw, @email, @kontakt, @login, @output, @typ);
         BEGIN
             PRINT ' twój login '+@login+' i hasło '+@output;
         END;
     END;
GO
```

b. procedura usuniecia klienta

```
CREATE PROCEDURE [dbo].[p_usun_klienta] @nazwisko VARCHAR(50)
AS
     BEGIN
         DECLARE @id_k INT;
         IF @nazwisko IS NULL
             BEGIN
                 PRINT 'BRAK ARGUMENTU';
                 RETURN;
             END;
         SET @id_k =
(
    SELECT id_klienta
```

```
    FROM klient
    WHERE nazwisko = @nazwisko
);
        IF @id_k IS NULL
            BEGIN
                PRINT 'nie ma takiego klienta';
            END;
        DELETE FROM klient
        WHERE nazwisko = @nazwisko;
```

c. procedura wyszukania klienta po nazwisku

```
CREATE PROCEDURE [dbo].[p_wyszukiwanie_klienta] @klient VARCHAR(50)
AS
     BEGIN
        IF @klient IS NULL
            BEGIN
                PRINT 'BRAK ARGUMENTU';
                RETURN;
            END;
        IF NOT EXISTS
(
    SELECT *
    FROM v_klient_dane
    WHERE nazwisko = @klient
)
            BEGIN
                PRINT 'klient ---'+STR(@klient)+'--- NIE ISTNIEJE';
                RETURN;
            END;
        SELECT *
        FROM v_klient_dane
        WHERE nazwisko = @klient;
     END
```

d. procedura zwraca ciag znaków do procedury dodaj klienta

```
CREATE PROC [dbo].[gen_haslo_login] @output VARCHAR(10) OUTPUT
AS
     DECLARE @zakres INT= 74;
     DECLARE @min INT= 48;
     DECLARE @char CHAR;
     DECLARE @dlug INT= 7;
     SET @output = '';
     WHILE @dlug > 0
        BEGIN
            SELECT @char = CHAR(ROUND(RAND() * @zakres + @min, 0));
            SET @output+=@char;
            SET @dlug = @dlug - 1;
        END;
     RETURN
```

e. procedura dodawania samochodu do bazy

```
CREATE PROC [dbo].[dodaj_samochod] @id_model    INT,
                                   @id_nadwozie INT,
                                   @cena        INT,
                                   @vin         VARCHAR(30),
                                   @przebieg    INT,
                                   @data_prod   INT,
                                   @kraj_p      CHAR(3),
                                   @ks          CHAR(1),
                                   @pojemnosc   INT,
                                   @paliwo      CHAR(1),
                                   @info        VARCHAR(50)
AS
     BEGIN
         IF @id_model IS NULL
            OR @id_nadwozie IS NULL
            OR @cena IS NULL
            OR @vin IS NULL
             BEGIN
                 PRINT 'BRAK ARGUMENTU';
                 RETURN;
             END;
         IF EXISTS
(
    SELECT id_samochod
    FROM samochody_na_sprzedaz
    WHERE vin = @vin
)
             BEGIN
                 PRINT 'samochod'+@vin+'jest już w systemie';
                 RETURN;
             END;
         INSERT INTO samochody_na_sprzedaz (id_model, id_nadwozie, cena, vin, przebieg,
                     data_produkcji, status, kraj_pochodzenia, ksiazka_serwisowa,
                     pojemnosc, paliwo, sam_informacje_dodatkowe)
         VALUES(@id_model, @id_nadwozie, @cena, @vin, @przebieg, @data_prod, 'W', @kraj_p,
                @ks, @pojemnosc, @paliwo, @info);
         BEGIN
             SET @id_model =
(
    SELECT id_samochod
    FROM samochody_na_sprzedaz
    WHERE vin = @vin
);
             PRINT 'dodano nowy samochod do bazy - nr. '+@id_model;
         END;
     END;
```

f. usuwanie samochodu z bazy

```
CREATE PROCEDURE [dbo].[p_usun_samochod] @id  INT,
                                         @vin VARCHAR(30)
AS
     BEGIN
         IF @id IS NULL AND @vin IS NULL
             BEGIN
                 PRINT 'BRAK ARGUMENTU';
                 RETURN;
             END;
         IF NOT EXISTS
(
    SELECT id_samochod
```

```
    FROM samochody_na_sprzedaz
    WHERE vin = @vin
          OR id_samochod = @id
)
            BEGIN
                PRINT 'brak samochodu w bazie';
                RETURN;
            END;
        DELETE FROM samochody_na_sprzedaz
        WHERE vin = @vin
              OR id_samochod = @id;
     END
```

g. procedura wyszukiwania samochodów po marce

```
CREATE PROCEDURE [dbo].[p_szukaj_marki] @producent_samochod VARCHAR(50)
AS
     BEGIN
         IF @producent_samochod IS NULL
             BEGIN
                 PRINT 'BRAK ARGUMENTU';
                 RETURN;
             END;
         IF NOT EXISTS
(
    SELECT *
    FROM v_samochody_do_sprzedania
    WHERE prod_nazwa = @producent_samochod
)
             BEGIN
                 PRINT 'BRAK TEGO MODELU '+@producent_samochod;
                 RETURN;
             END;
         SELECT *
         FROM v_samochody_do_sprzedania
         WHERE prod_nazwa = @producent_samochod;
     END
```

h. procedura sprzedaży

```
CREATE PROCEDURE dbo.p_sprzedaz
                 @id_s   INT ,
                 @id_k   INT ,
                 @cena   INT ,
                 @typ    CHAR(1) ,
                 @data   DATE ,
                 @status CHAR(1) ,
                 @id_sp  INT
AS
     BEGIN
         IF EXISTS ( SELECT *
                     FROM samochody_na_sprzedaz
                     WHERE id_samochod = @id_s
                   )
             BEGIN
                 IF EXISTS ( SELECT *
                             FROM klient
                             WHERE id_klienta = @id_k
                           )
                     BEGIN
                         IF ( SELECT status
                              FROM samochody_na_sprzedaz
                              WHERE id_samochod = @id_s
```

```
                        ) = 'W'
                         BEGIN
                             INSERT INTO sprzedane ( id_samochod , id_klient , cena ,
                                        typ_platnosci , data , status , id_sprzedawcy
                                                    )
                             VALUES ( @id_s , @id_k , @cena , @typ , @data , @status ,
                                      @id_sp
                                    );
                             UPDATE samochody_na_sprzedaz
                                    SET status = 'S'
                             WHERE id_samochod = @id_s;

                         END;
                END;
            END;
    END;
```

i. Usuwanie / dodanie rezerwacji

```
CREATE PROCEDURE [dbo].[p_zmiena_rezerwacji] @id_samochod INT,
                                             @status      CHAR(1),
                                             @id_klient   INT,
                                             @data        DATE
AS
    BEGIN
        DECLARE @stary_status CHAR(1);
        IF @id_klient IS NULL
           OR @status IS NULL
            BEGIN
                PRINT 'BRAK ARGUMENTU';
                RETURN;
            END;
        IF @status = 'R'
            BEGIN
                IF @data IS NULL
                   OR @id_samochod IS NULL
                   OR @data <= GETDATE()
                    BEGIN
                        PRINT 'BRAK ARGUMENTU';
                        RETURN;
                    END;
                IF NOT EXISTS
(
    SELECT *
    FROM v_samochody_do_sprzedania
    WHERE id_samochod = @id_samochod
)
                    BEGIN
                        PRINT 'BRAK samochodu '+STR(@id_samochod);
                        RETURN;
                    END;
                IF NOT EXISTS
(
    SELECT *
    FROM [dbo].[v_klient_dane]
    WHERE id_klienta = @id_klient
)
                    BEGIN
                        PRINT 'BRAK klienta o numerze '+STR(@id_samochod);
                        RETURN;
                    END;
                INSERT INTO rezerwacje
(id_klienta,
```

```
 id_samochodu,
 data_do
)
                VALUES
(@id_klient,
 @id_samochod,
 @data
);
                UPDATE samochody_na_sprzedaz
                  SET
                      status = @status
                WHERE id_samochod = @id_samochod;
            END;
            ELSE
        IF @status = 'W'
            BEGIN
                SET @id_samochod =
(
    SELECT id_samochodu
    FROM rezerwacje
    WHERE id_klienta = @id_klient
);
                UPDATE samochody_na_sprzedaz
                  SET
                      status = @status
                WHERE id_samochod = @id_samochod;
                DELETE FROM rezerwacje
                WHERE id_klienta = @id_klient;
            END;
            ELSE
            BEGIN
                PRINT 'W - usuwanie rezerwacji / R - dodanie rezerwacji ';
            END;
     END
```

TRIGERY

a. Trigery dla tabeli [klient]

-Triger sprawdza jaki typ użytkownika dodajemy (można dodać tylko ‘klienta’)

```
CREATE TRIGGER [dbo].[typ_klient] ON [dbo].[klient]
FOR INSERT
AS
     BEGIN
         DECLARE @operacja CHAR(1);
         SET @operacja =
(
    SELECT typ
    FROM inserted
);
         IF @operacja = 'D'
             BEGIN
                 RAISERROR('nie możesz dodać nowego dealera', 16, 1);
                 ROLLBACK TRANSACTION;
             END;
         IF @operacja = 'A'
             BEGIN
                 RAISERROR('nie możesz dodać nowego administratora', 16, 1);
                 ROLLBACK TRANSACTION;
             END;
         IF @operacja = 'S'
             BEGIN
```

```
                PRINT('dodano klienta');
            END;
            ELSE
            BEGIN
                RAISERROR('błędny typ klienta', 16, 1);
                ROLLBACK TRANSACTION;
            END;
    END;
```

- triger loguje wszystkie zmiany na tabeli

```
ALTER TRIGGER dbo.log_klient ON dbo.klient
AFTER INSERT , UPDATE , DELETE
AS
    BEGIN
        DECLARE @operacja CHAR(6);
        SET @operacja = CASE
                            WHEN EXISTS ( SELECT *
                                          FROM inserted
                                        )
                                 AND
                                 EXISTS ( SELECT *
                                          FROM deleted
                                        )
                            THEN 'Update'
                            WHEN EXISTS ( SELECT *
                                          FROM inserted
                                        )
                            THEN 'Insert'
                            WHEN EXISTS ( SELECT *
                                          FROM deleted
                                        )
                            THEN 'Delete'
                            ELSE NULL
                        END;
        IF @operacja = 'Delete'
            BEGIN
                INSERT INTO log_table ( log_1 , log_2 , tabela , operacjaA , operacjaB ,
                                        czas)
                       SELECT d.id_klienta , d.nazwisko+' '+d.imie , 'klient' , @operacja ,
USER_NAME() , GETDATE()
                       FROM deleted AS d
            END;
        IF @operacja = 'Insert'
            BEGIN
                INSERT INTO log_table ( log_1 , log_2 , tabela , operacjaA , operacjaB ,
                                         czas)
                       SELECT i.id_klienta , i.nazwisko+' '+i.imie , 'klient' , @operacja ,
USER_NAME() , GETDATE()
                       FROM inserted AS i
            END;
        IF @operacja = 'Update'
            BEGIN
                INSERT INTO log_table ( log_1 , log_2 , log_3 , log_4 , tabela , operacjaA
                                        , operacjaB , czas)
                       SELECT d.id_klienta , d.email+' '+d.kontakt+' '+d.login , i.email+'
                             '+i.kontakt , i.login , 'klient' , @operacja , USER_NAME() ,
                              GETDATE()
                       FROM deleted AS d , inserted AS i
            END;
    END;
```

- triger usuwa dane powiązane z klientem z tabel [klient_dane] i [klient_preferencje]

```
CREATE TRIGGER [dbo].[klient_delete_all] ON [dbo].[klient]
AFTER DELETE
AS
     BEGIN
         DECLARE @operacja INT;
         SET @operacja = ( SELECT id_klienta
                           FROM deleted
                         );
         IF EXISTS ( SELECT *
                     FROM klient_dane
                     WHERE id_klient = @operacja
                   )
             BEGIN
                 DELETE FROM klient_dane
                 WHERE id_klient = @operacja;
             END;
         IF EXISTS ( SELECT *
                     FROM klient_preferencje
                     WHERE id_klient = @operacja
                   )
             BEGIN
                 DELETE FROM klient_preferencje
                 WHERE id_klient = @operacja;
             END;
         ALTER TABLE klient CHECK CONSTRAINT ALL;
         ALTER TABLE klient_dane CHECK CONSTRAINT ALL;
         ALTER TABLE klient_preferencje CHECK CONSTRAINT ALL;
     END;
```

b. Triger loguje zmiany na tabeli [klient_dane]

```
ALTER TRIGGER [dbo].[log_klient_dane] ON [dbo].[klient_dane]
AFTER INSERT , UPDATE , DELETE
AS
     BEGIN
         DECLARE @operacja CHAR(6);
         SET @operacja = CASE
                             WHEN EXISTS ( SELECT *
                                           FROM inserted
                                         )
                                  AND
                                  EXISTS ( SELECT *
                                           FROM deleted
                                         )
                             THEN 'Update'
                             WHEN EXISTS ( SELECT *
                                           FROM inserted
                                         )
                             THEN 'Insert'
                             WHEN EXISTS ( SELECT *
                                           FROM deleted
                                         )
                             THEN 'Delete'
                             ELSE NULL
                         END;
         IF @operacja = 'Delete'
             BEGIN
                 INSERT INTO log_table ( log_1 , log_2 , tabela , operacjaA , operacjaB ,
                                         czas
                                       )
```

```
                        SELECT d.id_klient , d.id_adres , 'klient_dane' , @operacja ,
                                USER_NAME() , GETDATE()
                        FROM deleted AS d
            END;
        IF @operacja = 'Insert'
            BEGIN
                INSERT INTO log_table ( log_1 , log_2 , tabela , operacjaA , operacjaB ,
                                        czas
                                      )
                        SELECT i.id_klient , i.id_adres , 'klient_dane' , @operacja ,
USER_NAME() , GETDATE()
                        FROM inserted AS i
            END;
        IF @operacja = 'Update'
            BEGIN
                INSERT INTO log_table ( log_1 , log_2 , log_3 , log_4 , tabela , operacjaA
                                        , operacjaB , czas
                                      )
                        SELECT d.id_klient , i.id_adres , i.adres_linia_1 , d.adres_linia_1
                                , 'klient_dane' , @operacja , USER_NAME() , GETDATE()
                        FROM deleted AS d , inserted AS i
            END;
    END;
```

c. Triger usuwa dane powiązane z samochodem w tabelach [rezerwacje] i [wyposażenie]

```
ALTER TRIGGER [dbo].[del_wyposazenie] ON [dbo].[samochody_na_sprzedaz]
AFTER DELETE
AS
    BEGIN
        DECLARE @operacja INT;
        SET @operacja = ( SELECT id_samochod
                          FROM deleted
                        );
        IF EXISTS ( SELECT *
                    FROM rezerwacje
                    WHERE id_samochodu = @operacja
                  )
            BEGIN
                DELETE FROM rezerwacje
                WHERE id_samochodu = @operacja;
            END;
        IF EXISTS ( SELECT *
                    FROM powiazanie_wyposazenie_samochod
                    WHERE id_samochod = @operacja
                  )
            BEGIN
                DELETE FROM powiazanie_wyposazenie_samochod
                WHERE id_samochod = @operacja;
            END;
        ALTER TABLE rezerwacje CHECK CONSTRAINT ALL;
        ALTER TABLE powiazanie_wyposazenie_samochod CHECK CONSTRAINT ALL;
    END;
```