Michał Szałkowski
PSISK NS

Auto-Handel:

- ***Klienci:*** id_klienta, nazwisko, imie, miasto, ulica, kod_pocztowy, nr_telefonu
- ***Sprzedawcy:*** id_sprzedawcy, id_komisu, nazwisko, imie, stanowisko, data_zatrudnienia, pensja, nr_telefonu
- ***Komis:*** id_komisu, miasto, ulica, kod_pocztowy
- ***Samochody:*** id_samochodu, id_komisu, marka, model_auta, rok_produkcji, pojemnosc_cm3, rodzaj_paliwa, moc_km, przebieg, nr_vin, kwota_do_zaplaty, czy_sprzedany
- ***Zamowienia:*** id_zamowienia, data_zamowienia, id_sprzedawcy, id_klienta, id_samochodu
- ***Platnosc:*** id_platnosci, id_zamowienia, kwota_zaplacona

1. Przygotowanie odpowiedniej struktury bazy danych na wybrany temat *[DONE]*

Klienci: id_klienta, nazwisko, imie, miasto, ulica, kod_pocztowy, nr_telefonu

```
CREATE TABLE Klienci
(id_klienta NUMBER(10) PRIMARY KEY,
nazwisko VARCHAR(30) NOT NULL,
imie VARCHAR(20) NOT NULL,
miasto VARCHAR(20) NOT NULL,
ulica VARCHAR(20) NOT NULL,
kod_pocztowy VARCHAR(10) NOT NULL,
nr_telefonu NUMBER(9))
```

Sprzedawcy: id_sprzedawcy, id_komisu, nazwisko, imie, email, stanowisko, data_zatrudnienia, pensja, nr_telefonu

```
CREATE TABLE Sprzedawcy
(id_sprzedawcy NUMBER(10) PRIMARY KEY,
id_komisu NUMBER(10),
nazwisko VARCHAR(30) NOT NULL,
imie VARCHAR(20) NOT NULL,
email VARCHAR(100),
stanowisko VARCHAR(20) NOT NULL,
data_zatrudnienia DATE,
pensja NUMBER(10,2),
nr_telefonu NUMBER(9))
```

Komis: id_komisu, miasto, ulica, kod_pocztowy

```
CREATE TABLE Komis
(id_komisu NUMBER(10) PRIMARY KEY,
miasto VARCHAR(20) NOT NULL,
ulica VARCHAR(20) NOT NULL,
kod_pocztowy VARCHAR(10) NOT NULL)
```

Samochody: id_samochodu, id_komisu, marka, model_auta, rok_produkcji, pojemnosc_cm3, rodzaj_paliwa, moc_km, przebieg, nr_vin, kwota_do_zaplaty, czy_sprzedany

```
CREATE TABLE Samochody
(id_samochodu NUMBER(10) PRIMARY KEY,
id_komisu NUMBER(10),
marka VARCHAR(30) NOT NULL,
model_auta VARCHAR(30) NOT NULL,
```

```
rok_produkcji NUMBER(10),
pojemnosc_cm3 NUMBER(10),
rodzaj_paliwa VARCHAR(30) NOT NULL,
moc_km NUMBER(10),
przebieg NUMBER(10),
nr_vin VARCHAR(17) NOT NULL,
kwota_do_zaplaty NUMBER(10,2),
czy_sprzedany NUMBER(1,0))
```

Zamowienia: id_zamowienia, data_zamowienia, id_sprzedawcy, id_klienta, id_samochodu

```
CREATE TABLE Zamowienia
(id_zamowienia NUMBER(10) PRIMARY KEY,
data_zamowienia DATE,
id_sprzedawcy NUMBER(10),
id_klienta NUMBER(10),
id_samochodu NUMBER(10))
```

Platnosc: id_platnosci, id_zamowienia, kwota_zaplacona

```
CREATE TABLE Platnosc
(id_platnosci NUMBER(10) PRIMARY KEY,
id_zamowienia NUMBER(10),
kwota_zaplacona NUMBER(10,2))
```

Klucze obce:

```
ALTER TABLE Sprzedawcy ADD FOREIGN KEY(id_komisu) REFERENCES Komis(id_komisu);
ALTER TABLE Samochody ADD FOREIGN KEY(id_komisu) REFERENCES Komis(id_komisu);
ALTER TABLE Zamowienia ADD FOREIGN KEY(id_sprzedawcy) REFERENCES
Sprzedawcy(id_sprzedawcy);
ALTER TABLE Zamowienia ADD FOREIGN KEY(id_klienta) REFERENCES
Klienci(id_klienta);
ALTER TABLE Zamowienia ADD FOREIGN KEY(id_samochodu) REFERENCES
Samochody(id_samochodu);
ALTER TABLE Platnosc ADD FOREIGN KEY(id_zamowienia) REFERENCES
Zamowienia(id_zamowienia);
```

2. Ładowanie do bazy przykładowych danych *[DONE]*

Uzupełnienie rekordów --- KLIENCI:

```
INSERT INTO Klienci (id_klienta, nazwisko, imie, miasto, ulica, kod_pocztowy,
nr_telefonu)
VALUES (1, 'Kowalski', 'Jan', 'Olsztyn', 'Sloneczna 1', '10-166', 999777111);

INSERT INTO Klienci (id_klienta, nazwisko, imie, miasto, ulica, kod_pocztowy,
nr_telefonu)
VALUES (2, 'Nowak', 'Anna', 'Olsztyn', 'Dluga 2', '10-166', 111222333);

INSERT INTO Klienci (id_klienta, nazwisko, imie, miasto, ulica, kod_pocztowy,
nr_telefonu)
VALUES (3, 'Wysoki', 'Adam', 'Ilawa', 'Morska 3', '14-200', 222333444);

INSERT INTO Klienci (id_klienta, nazwisko, imie, miasto, ulica, kod_pocztowy,
nr_telefonu)
VALUES (4, 'Niski', 'Sebastian', 'Ostroda', 'Polna 4', '14-100', 333444555);

INSERT INTO Klienci (id_klienta, nazwisko, imie, miasto, ulica, kod_pocztowy,
nr_telefonu)
VALUES (5, 'Wisniewski', 'Adam', 'Ilawa', 'Warminska 5', '14-200', 444555666);
```

Uzupelnienie rekordów --- Komis:

```
INSERT INTO Komis (id_komisu, miasto, ulica, kod_pocztowy)
VALUES (1, 'Olsztyn', 'Sloneczna 54', '10-710')

INSERT INTO Komis (id_komisu, miasto, ulica, kod_pocztowy)
VALUES (2, 'Ilawa', 'Wiejska 1', '14-200')

INSERT INTO Komis (id_komisu, miasto, ulica, kod_pocztowy)
VALUES (3, 'Ostroda', 'Nowa 3', '14-100')
```

Uzupelnienie rekordów --- Sprzedawcy:

```
INSERT INTO Sprzedawcy (id_sprzedawcy, id_komisu, nazwisko, imie, email,
stanowisko, data_zatrudnienia, pensja, nr_telefonu)
VALUES (1, 1, 'Znawca', 'Adam', '', 'Kierownik', '01/04/21', 7000, 777222777)

INSERT INTO Sprzedawcy (id_sprzedawcy, id_komisu, nazwisko, imie, email,
stanowisko, data_zatrudnienia, pensja, nr_telefonu)
VALUES (2, 2, 'Myslacy', 'Krzysztof', '', 'Doradca Klienta', '02/04/21', 5000,
555444777)

INSERT INTO Sprzedawcy (id_sprzedawcy, id_komisu, nazwisko, imie, email,
stanowisko, data_zatrudnienia, pensja, nr_telefonu)
VALUES (3, 3, 'Znany', 'Marcin', '', 'Doradca Klienta', '03/04/21', 3000, 0)
```

Uzupelnienie rekordów --- Samochody w komisie 1:

```
INSERT INTO Samochody
(
id_samochodu, id_komisu, marka, model_auta, rok_produkcji, pojemnosc_cm3,
rodzaj_paliwa, moc_km, przebieg, nr_vin, kwota_do_zaplaty, czy_sprzedany
)
VALUES (1, 1, 'Audi', 'A3', 2005, 2000, 'Benzyna', 140, 150000,
'WAUZZZ8P666A0444683', 9900, 0)


INSERT INTO Samochody
(
id_samochodu, id_komisu, marka, model_auta, rok_produkcji, pojemnosc_cm3,
rodzaj_paliwa, moc_km, przebieg, nr_vin, kwota_do_zaplaty, czy_sprzedany
)
VALUES (2, 1, 'BMW', 'M3', 2015, 2979, 'Benzyna', 550, 99756,
'WBS3C9C53FP804574', 174900, 0)


INSERT INTO Samochody
(
id_samochodu, id_komisu, marka, model_auta, rok_produkcji, pojemnosc_cm3,
rodzaj_paliwa, moc_km, przebieg, nr_vin, kwota_do_zaplaty, czy_sprzedany
)
VALUES (3, 1, 'Opel', 'Vectra', 2006, 1796, 'Benzyna', 140, 130000,
'XXXXXXXXXXXXXXXXX', 13900, 0)
```

```
INSERT INTO Samochody
(
id_samochodu, id_komisu, marka, model_auta, rok_produkcji, pojemnosc_cm3,
rodzaj_paliwa, moc_km, przebieg, nr_vin, kwota_do_zaplaty, czy_sprzedany
)
VALUES (4, 1, 'Opel', 'Corsa', 2008, 1200, 'Benzyna', 80, 170412, 'X', 11900,
0)


INSERT INTO Samochody
(
id_samochodu, id_komisu, marka, model_auta, rok_produkcji, pojemnosc_cm3,
rodzaj_paliwa, moc_km, przebieg, nr_vin, kwota_do_zaplaty, czy_sprzedany
)
VALUES (5, 1, 'Opel', 'Insignia', 2012, 1956, 'Benzyna', 194, 221000,
'W0LGS8EN5C1096202', 29900, 0)

Uzupelnienie rekordów --- Samochody w komisie 2:
INSERT INTO Samochody
(
id_samochodu, id_komisu, marka, model_auta, rok_produkcji, pojemnosc_cm3,
rodzaj_paliwa, moc_km, przebieg, nr_vin, kwota_do_zaplaty, czy_sprzedany
)
VALUES (6, 2, 'Fiat', 'Doblo II', 2007, 1910, 'Benzyna', 120, 218001,
'ZFA22200005424823', 14300, 0)

INSERT INTO Samochody
(
id_samochodu, id_komisu, marka, model_auta, rok_produkcji, pojemnosc_cm3,
rodzaj_paliwa, moc_km, przebieg, nr_vin, kwota_do_zaplaty, czy_sprzedany
)
VALUES (7, 2, 'Volkswagen', 'Polo', 2011, 1198, 'Benzyna', 69, 159500,
'WVWZZZ6RZBY294822', 19950, 0)

INSERT INTO Samochody
(
id_samochodu, id_komisu, marka, model_auta, rok_produkcji, pojemnosc_cm3,
rodzaj_paliwa, moc_km, przebieg, nr_vin, kwota_do_zaplaty, czy_sprzedany
)
VALUES (8, 2, 'Volkswagen', 'Golf VIII', 2020, 1968, 'Diesel', 115, 1,
'WVWZZZ6RZBY294825', 102900, 0)

INSERT INTO Samochody
(
id_samochodu, id_komisu, marka, model_auta, rok_produkcji, pojemnosc_cm3,
rodzaj_paliwa, moc_km, przebieg, nr_vin, kwota_do_zaplaty, czy_sprzedany
)
VALUES (9, 3, 'Volkswagen', 'Passat', 2020, 1986, 'Diesel', 115, 194300,
'WVWZZZ6RZBY294825', 48900, 0)
```

```
Uzupelnienie rekordów --- Samochody w komisie 3:
INSERT INTO Samochody
(
id_samochodu, id_komisu, marka, model_auta, rok_produkcji, pojemnosc_cm3,
rodzaj_paliwa, moc_km, przebieg, nr_vin, kwota_do_zaplaty, czy_sprzedany
)
VALUES (10, 3, 'Volvo', 'V60', 2016, 1969, 'Benzyna', 150, 123654,
'USYHC516AAL206925', 51800, 0)

INSERT INTO Samochody
(
id_samochodu, id_komisu, marka, model_auta, rok_produkcji, pojemnosc_cm3,
rodzaj_paliwa, moc_km, przebieg, nr_vin, kwota_do_zaplaty, czy_sprzedany
)
VALUES (11, 3, 'Volvo', 'V40', 2013, 1984, 'Benzyna', 150, 180000,
'YV1MZ515R31042023', 41500, 0)

INSERT INTO Samochody
(
id_samochodu, id_komisu, marka, model_auta, rok_produkcji, pojemnosc_cm3,
rodzaj_paliwa, moc_km, przebieg, nr_vin, kwota_do_zaplaty, czy_sprzedany
)
VALUES (12, 3, 'Seat', 'Altea', 2009, 1790, 'Benzyna', 160, 1420005,
'VSSZZZSPZ9R002604', 157999, 0)

Uzupelnienie rekordów --- Zamowienia:
INSERT INTO Zamowienia
(
id_zamowienia, data_zamowienia, id_sprzedawcy, id_klienta, id_samochodu
)
VALUES (1, '05/04/21', 1, 2, 3)

INSERT INTO Zamowienia
(
id_zamowienia, data_zamowienia, id_sprzedawcy, id_klienta, id_samochodu
)
VALUES (2, '05/04/21', 1, 1, 2)

INSERT INTO Zamowienia
(
id_zamowienia, data_zamowienia, id_sprzedawcy, id_klienta, id_samochodu
)
VALUES (3, '06/04/21', 1, 3, 1)

INSERT INTO Zamowienia
(
id_zamowienia, data_zamowienia, id_sprzedawcy, id_klienta, id_samochodu
)
VALUES (4, '06/04/21', 2, 4, 4)

INSERT INTO Zamowienia
(
id_zamowienia, data_zamowienia, id_sprzedawcy, id_klienta, id_samochodu
)
```

```
VALUES (5, '06/04/21', 3, 5, 5)
```

Uzupelnienie rekordów --- Platnosc

```
INSERT INTO Platnosc
(
id_platnosci, id_zamowienia, kwota_zaplacona
)
VALUES (1, 5, 29900)
```

```
INSERT INTO Platnosc
(
id_platnosci, id_zamowienia, kwota_zaplacona
)
VALUES (2, 4, 11900)
```

```
INSERT INTO Platnosc
(
id_platnosci, id_zamowienia, kwota_zaplacona
)
VALUES (3, 3, 9900)
```

```
INSERT INTO Platnosc
(
id_platnosci, id_zamowienia, kwota_zaplacona
)
VALUES (4, 2, 174900)
```

```
INSERT INTO Platnosc
(
id_platnosci, id_zamowienia, kwota_zaplacona
)
VALUES (5, 1, 13900)
```

3. Procedury, funkcje, wyzwalacze obsługujące bazę [DONE]
a. Dodawanie, usuwanie, aktualizacja rekordów

3.1.

```
[==============================]
[PROCEDURA DODAJACA SPRZEDAWCE]
[==============================]
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE dodaj_sprzedawce (
p_id_sprzedawcy SPRZEDAWCY.id_sprzedawcy%TYPE,
p_id_komisu SPRZEDAWCY.id_komisu%TYPE,
p_nazwisko SPRZEDAWCY.nazwisko%TYPE,
p_imie SPRZEDAWCY.imie%TYPE,
p_email SPRZEDAWCY.email%TYPE,
p_stanowisko SPRZEDAWCY.stanowisko%TYPE,
p_data_zatrudnienia SPRZEDAWCY.data_zatrudnienia%TYPE,
p_pensja SPRZEDAWCY.pensja%TYPE,
p_nr_telefonu SPRZEDAWCY.nr_telefonu%TYPE
```

```
)
IS
BEGIN
INSERT INTO Sprzedawcy
(id_sprzedawcy, id_komisu, nazwisko, imie, email, stanowisko,
data_zatrudnienia, pensja, nr_telefonu)
VALUES
(p_id_sprzedawcy, p_id_komisu, p_nazwisko, p_imie, p_email, p_stanowisko,
p_data_zatrudnienia, p_pensja, p_nr_telefonu);
dbms_output.put_line('Sprzedawca zostal dodany pomyslnie!');
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
      dbms_output.PUT_LINE('[Wyjatek] Istnieje juz sprzedawca o podanym
identyfikatorze!');
      WHEN OTHERS THEN
      dbms_output.PUT_LINE('[Wyjatek] Nieoczekiwany blad');
END;

DECLARE
BEGIN
dodaj_sprzedawce(4, 1, 'Nowy', 'Jan', 'janek@poczta.pl', 'Handlarz',
'05/04/21', 6000, 999222999);
END;

3.2.
[==============================]
[PROCEDURA USUWAJĄCA SPRZEDAWCE]
[==============================]
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE usun_sprzedawce (
p_id_sprzedawcy SPRZEDAWCY.id_sprzedawcy%TYPE
)
IS
nie_usunieto_sprzedawcy EXCEPTION;
BEGIN
DELETE FROM SPRZEDAWCY WHERE id_sprzedawcy = p_id_sprzedawcy ;
IF SQL%ROWCOUNT = 0 THEN
raise nie_usunieto_sprzedawcy;
END IF;
dbms_output.put_line('Usunieto sprzedawce z tabeli SPRZEDAWCY');
EXCEPTION
    WHEN nie_usunieto_sprzedawcy THEN
      dbms_output.PUT_LINE('[Wyjatek] Brak sprzedawcy o podanym id!');
      WHEN OTHERS THEN
      dbms_output.PUT_LINE('[Wyjatek] Nieoczekiwany blad');
END;


DECLARE
BEGIN
usun_sprzedawce(4);
END;
```

3.3.

```
[========================================]
[PROCEDURA AKTUALIZUJACA PENSJE SPRZEDAWCY]
[========================================]
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE aktualizuj_pensje_sprzedawcy (
p_id_sprzedawcy SPRZEDAWCY.id_sprzedawcy%TYPE,
p_pensja SPRZEDAWCY.pensja%TYPE
)
IS
nie_zaktualizowano_sprzedawcy EXCEPTION;
BEGIN
UPDATE SPRZEDAWCY SET pensja = p_pensja WHERE id_sprzedawcy = p_id_sprzedawcy
;
IF SQL%ROWCOUNT = 0 THEN
raise nie_zaktualizowano_sprzedawcy;
END IF;
dbms_output.put_line('Zaktualizowano sprzedawce');
EXCEPTION
    WHEN nie_zaktualizowano_sprzedawcy THEN
      dbms_output.PUT_LINE('[Wyjatek] Brak sprzedawcy o podanym id!');
      WHEN OTHERS THEN
      dbms_output.PUT_LINE('[Wyjatek] Nieoczekiwany blad');
END;

DECLARE
BEGIN
aktualizuj_pensje_sprzedawcy(4, 7000);
END;
```

3.4.

```
[============================================]
[FUNKCJA Z PARAMETRAMI GENERUJACA ADRES E-MAIL]
[============================================]
SET SERVEROUTPUT ON;

create or replace function wygenerujmail (
p_id_sprzedawcy SPRZEDAWCY.id_sprzedawcy%TYPE
)
RETURN VARCHAR2 AS v_mail VARCHAR2(100);
BEGIN
SELECT lower(nazwisko)||'.'||lower(imie)||'@'||'auto-handel-zsbd.pl' INTO
v_mail FROM sprzedawcy WHERE id_sprzedawcy = p_id_sprzedawcy;
RETURN v_mail;
END;

DECLARE
v_mail VARCHAR2(100);
BEGIN
v_mail := wygenerujmail(3);
dbms_output.PUT_LINE(v_mail);
END;
```

3.5.

```
[================================================================]
[PROCEDURA AKTUALIZUJACA ADRES EMAIL WSKAZANEMU PRACOWNIKOWI - WYKORZYSTUJACA
FUNKCJE GENEROWANIA MAILA]
[================================================================]
SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE ZAKTUALIZUJ_MAILA_SPRZEDAWCY (p_id_sprzedawcy
SPRZEDAWCY.id_sprzedawcy%TYPE)
IS
no_mail_updated EXCEPTION;
wygenerowany_mail SPRZEDAWCY.email%TYPE;
BEGIN
wygenerowany_mail := wygenerujmail(p_id_sprzedawcy);
UPDATE SPRZEDAWCY SET email = wygenerowany_mail WHERE id_sprzedawcy =
p_id_sprzedawcy ;
IF SQL%ROWCOUNT = 0 THEN
raise no_mail_updated;
END IF;
      dbms_output.PUT_LINE('Adres email zaktualizowano pomyslnie na:
'||wygenerowany_mail);
EXCEPTION
    WHEN no_mail_updated THEN
    dbms_output.put_line('Brak wierszy o podanym id');
END;

DECLARE
BEGIN
ZAKTUALIZUJ_MAILA_SPRZEDAWCY(1);
END;
```

3.6.

```
[================================================================]
[FUNKCJA SPRAWDZAJACA CZY WPROWADZONY VIN SAMOCHODU JEST POPRAWNY]
[================================================================]
SET SERVEROUTPUT ON;

create or replace function SPRAWDZ_POPRAWNOSC_VIN (p_nr_vin
SAMOCHODY.nr_vin%TYPE)
RETURN VARCHAR2 AS v_czy_poprawny NUMBER(1,0);
BEGIN
IF LENGTH(p_nr_vin) =  17 THEN
v_czy_poprawny := 1;
ELSE
v_czy_poprawny := 0;
END IF;
RETURN v_czy_poprawny;
END;

DECLARE
v_czy_poprawny NUMBER(1,0);
BEGIN
v_czy_poprawny := SPRAWDZ_POPRAWNOSC_VIN('1234567891ASDFGHJ');
```

```
dbms_output.PUT_LINE(v_czy_poprawny);
if v_czy_poprawny = 1 THEN
dbms_output.PUT_LINE('poprawny nr vin');
else
dbms_output.PUT_LINE('NIEPOPRAWNY nr vin');
end if;
END;

3.7.
[================================================================================
]
[PROCEDURA DODAJACA NOWY SAMOCHOD -- WYKORZYSTUJACA FUNKCJE SPRAWDZANIA NR
VIN]
[================================================================================
]
SET SERVEROUTPUT ON;
CREATE SEQUENCE samochod_id_seq
MINVALUE 1
START WITH 13
INCREMENT BY 1
CACHE 20;

CREATE OR REPLACE PROCEDURE dodaj_samochod (
p_id_komisu SAMOCHODY.id_komisu%TYPE,
p_marka SAMOCHODY.marka%TYPE,
p_model_auta SAMOCHODY.model_auta%TYPE,
p_rok_produkcji SAMOCHODY.rok_produkcji%TYPE,
p_pojemnosc_cm3 SAMOCHODY.pojemnosc_cm3%TYPE,
p_rodzaj_paliwa SAMOCHODY.rodzaj_paliwa%TYPE,
p_moc_km SAMOCHODY.moc_km%TYPE,
p_przebieg SAMOCHODY.przebieg%TYPE,
p_nr_vin SAMOCHODY.nr_vin%TYPE,
p_kwota_do_zaplaty SAMOCHODY.kwota_do_zaplaty%TYPE,
p_czy_sprzedany SAMOCHODY.czy_sprzedany%TYPE
)
IS
BEGIN
IF SPRAWDZ_POPRAWNOSC_VIN(p_nr_vin) = 1 THEN
INSERT INTO Samochody
(id_samochodu, id_komisu, marka, model_auta, rok_produkcji, pojemnosc_cm3,
rodzaj_paliwa, moc_km, przebieg, nr_vin, kwota_do_zaplaty, czy_sprzedany)
VALUES
(samochod_id_seq.nextval, p_id_komisu, p_marka, p_model_auta, p_rok_produkcji,
p_pojemnosc_cm3, p_rodzaj_paliwa, p_moc_km, p_przebieg, p_nr_vin,
p_kwota_do_zaplaty, p_czy_sprzedany);
dbms_output.put_line('Samochod zostal dodany pomyslnie!');
ELSE
dbms_output.put_line('Niepoprawny NR VIN!');
END IF;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
      dbms_output.PUT_LINE('[Wyjatek] Istnieje juz samochod o podanym
identyfikatorze!');
      WHEN OTHERS THEN
      dbms_output.PUT_LINE('[Wyjatek] Nieoczekiwany blad');
END;
```

```
DECLARE
BEGIN
dodaj_samochod(2, 'Volkswagen', 'Golf VIII', 2020, 1968, 'Diesel', 115, 1,
'WV2WZ2ZRZBY294825', 102900, 0);
END;
```

3.8.

```
[=================================================================]
[FUNKCJA SPRAWDZAJACA CZY WPROWADZONY NR TELEFONU KLIENTA JEST POPRAWNY]
[=================================================================]
SET SERVEROUTPUT ON;

create or replace function SPRAWDZ_POPRAWNOSC_NR_TELEFONU (p_nr_tel
KLIENCI.nr_telefonu%TYPE)
RETURN VARCHAR2 AS v_czy_poprawny NUMBER(1,0);
BEGIN
IF LENGTH(p_nr_tel) =  9 THEN
v_czy_poprawny := 1;
ELSE
v_czy_poprawny := 0;
END IF;
RETURN v_czy_poprawny;
END;

DECLARE
v_czy_poprawny NUMBER(1,0);
BEGIN
v_czy_poprawny := SPRAWDZ_POPRAWNOSC_NR_TELEFONU('123456789');
dbms_output.PUT_LINE(v_czy_poprawny);
if v_czy_poprawny = 1 THEN
dbms_output.PUT_LINE('poprawny nr telefonu');
else
dbms_output.PUT_LINE('NIEPOPRAWNY nr telefonu');
end if;
END;
```

3.9.

```
[=============================================================================]
[PROCEDURA DODAJACA NOWEGO KLIENTA -- WYKORZYSTUJACA FUNKCJE SPRAWDZANIA NR
TELEFONU]
[=============================================================================]

SET SERVEROUTPUT ON;
CREATE SEQUENCE id_klienta_seq
MINVALUE 1
START WITH 6
INCREMENT BY 1
CACHE 20;

CREATE OR REPLACE PROCEDURE dodaj_klienta (
p_nazwisko KLIENCI.nazwisko%TYPE,
p_imie KLIENCI.imie%TYPE,
p_miasto KLIENCI.miasto%TYPE,
p_ulica KLIENCI.ulica%TYPE,
p_kod_pocztowy KLIENCI.kod_pocztowy%TYPE,
p_nr_telefonu KLIENCI.nr_telefonu%TYPE
)
```

```
IS
BEGIN
IF SPRAWDZ_POPRAWNOSC_NR_TELEFONU(p_nr_telefonu) = 1 THEN
INSERT INTO KLIENCI
(id_klienta, nazwisko, imie, miasto, ulica, kod_pocztowy, nr_telefonu)
VALUES
(id_klienta_seq.nextval, p_nazwisko, p_imie, p_miasto, p_ulica,
p_kod_pocztowy, p_nr_telefonu);
dbms_output.put_line('Klient zostal dodany pomyslnie!');
ELSE
dbms_output.put_line('Niepoprawny NR TELEFONU!');
END IF;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
      dbms_output.PUT_LINE('[Wyjatek] Istnieje juz klient o podanym
identyfikatorze!');
      WHEN OTHERS THEN
      dbms_output.PUT_LINE('[Wyjatek] Nieoczekiwany blad');
END;

DECLARE
BEGIN
dodaj_klienta('Wysoki', 'Jan', 'Warszawa', 'Wiejska 1', '00-500', 123456789);
END;

DECLARE
BEGIN
dodaj_klienta('Wysoki', 'Jan', 'Warszawa', 'Wiejska 1', '00-500', 123);
END;
[=========================================]
[=========================================]
[=========================================]
[=========================================]

3.10.
[=========================================]
[==== HISTORIA - ARCHIWUM samochodow =====]
[=========================================]

SET SERVEROUTPUT ON;
CREATE SEQUENCE id_historia_seq
MINVALUE 1
START WITH 1
INCREMENT BY 1
CACHE 20;

CREATE TABLE Samochody_historia
(
id_historia NUMBER(10) PRIMARY KEY,
hist_id_samochodu NUMBER(10),
hist_id_komisu NUMBER(10),
hist_marka VARCHAR(30) NOT NULL,
hist_model_auta VARCHAR(30) NOT NULL,
hist_rok_produkcji NUMBER(10),
hist_pojemnosc_cm3 NUMBER(10),
hist_rodzaj_paliwa VARCHAR(30) NOT NULL,
hist_moc_km NUMBER(10),
```

```
hist_przebieg NUMBER(10),
hist_nr_vin VARCHAR(17) NOT NULL,
hist_kwota_do_zaplaty NUMBER(10,2),
hist_czy_sprzedany NUMBER(1,0))

CREATE OR REPLACE TRIGGER update_samochody
before update
on samochody
begin
dbms_output.put_line('[Wyzwalacz] Zaktualizowano wiersz w tabeli samochody');
end;

CREATE OR REPLACE TRIGGER insert_samochody
before insert
on samochody
begin
dbms_output.put_line('[Wyzwalacz] Wstawiono wiersz w tabeli samochody');
end;

CREATE OR REPLACE TRIGGER delete_samochody
after delete
on samochody
FOR EACH ROW
begin
INSERT INTO Samochody_historia
(
id_historia, hist_id_samochodu, hist_id_komisu, hist_marka, hist_model_auta,
hist_rok_produkcji, hist_pojemnosc_cm3, hist_rodzaj_paliwa, hist_moc_km,
hist_przebieg, hist_nr_vin, hist_kwota_do_zaplaty, hist_czy_sprzedany
)
VALUES
(id_historia_seq.nextval, :OLD.id_samochodu, :OLD.id_komisu, :OLD.marka,
:OLD.model_auta, :OLD.rok_produkcji, :OLD.pojemnosc_cm3, :OLD.rodzaj_paliwa,
:OLD.moc_km, :OLD.przebieg, :OLD.nr_vin, :OLD.kwota_do_zaplaty,
:OLD.czy_sprzedany);
dbms_output.put_line('[Wyzwalacz] Usunieto wiersz z tabeli samochody');
end;

delete samochody where id_samochodu = 14;
update samochody set model_auta='Altea2' where model_auta='Aleta'
==== END HISTORIA - ARCHIWUM samochodow ===

3.11.
[=========================================]
[=============== LOGI ====================]
[=========================================]

CREATE SEQUENCE id_log_seq
MINVALUE 1
START WITH 1
INCREMENT BY 1
CACHE 20;

CREATE TABLE Samochody_logi
(
id_log NUMBER(10) PRIMARY KEY,
zmiana VARCHAR(20),
```

```
data_operacji TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
hist_id_samochodu NUMBER(10))

[==================================]
[====WYZWALACZ UPDATE SAMOCHODY====]
[==================================]
set serveroutput on;
CREATE OR REPLACE TRIGGER update_samochody
after update
on samochody
for each row
begin
INSERT INTO samochody_logi(id_log, zmiana, hist_id_samochodu)
VALUES (id_log_seq.nextval, 'update', :OLD.id_samochodu);
dbms_output.put_line('[Wyzwalacz] Zaktualizowano wiersz w tabeli samochody');
end;

update samochody set model_auta='Altea2' where model_auta='Altea'


[========================================]
[========WYZWALACZ INSERT SAMOCHODY=======]
[========================================]
set serveroutput on;
CREATE OR REPLACE TRIGGER insert_samochody
after insert
on samochody
for each row
begin
INSERT INTO samochody_logi(id_log, zmiana, hist_id_samochodu)
VALUES (id_log_seq.nextval, 'insert', :OLD.id_samochodu);
dbms_output.put_line('[Wyzwalacz] Dodano wiersz w tabeli samochody');
end;


INSERT INTO Samochody
(
id_samochodu, id_komisu, marka, model_auta, rok_produkcji, pojemnosc_cm3,
rodzaj_paliwa, moc_km, przebieg, nr_vin, kwota_do_zaplaty, czy_sprzedany
)
VALUES (15, 3, 'Seat', 'Altea', 2009, 1790, 'Benzyna', 160, 1420005,
'VSSZZZSPZ9R002604', 157999, 0)

==== END LOGI ===

4. Procedury, funkcje, wyzwalacze tworzące podsumowania
a. Zestawienia miesięczne, kwartalne, roczne, w zależności  od różnych
parametrów – zapisywane w bazie danych [DONE]

4.1.
[=============================================================================]
[===ZESTAWIENIE TWORZACE=====================================================]
[===PODSUMOWANIE SPRZEDANYCH AUT PRZEZ PRACOWNIKA NR 1 W MIESIACU KWIETNIU===]
[=============================================================================]
CREATE VIEW zestawienie1 AS
SELECT
ZAMOWIENIA.id_zamowienia,
```

```
ZAMOWIENIA.data_zamowienia,
SAMOCHODY.marka,
SAMOCHODY.model_auta,
SPRZEDAWCY.imie AS imie_sprzedawcy,
SPRZEDAWCY.nazwisko AS nazwisko_sprzedawcy
FROM ZAMOWIENIA
INNER JOIN SAMOCHODY on zamowienia.id_samochodu = samochody.id_samochodu
INNER JOIN SPRZEDAWCY on ZAMOWIENIA.id_sprzedawcy = SPRZEDAWCY.id_sprzedawcy
WHERE data_zamowienia BETWEEN '01/04/21' AND '30/04/21' AND
SPRZEDAWCY.id_sprzedawcy='1'
[================================================================================
====================]
```

4.2.

```
[================================================================================]
[===ZESTAWIENIE TWORZACE=========================================================]
[===PODSUMOWANIE SPRZEDANYCH AUT PRZEZ WSZYSTKICH PRACOWNIKÓW==============]
[================================================================================]
CREATE VIEW zestawienie2 AS
SELECT
SPRZEDAWCY.id_sprzedawcy,
SPRZEDAWCY.imie AS imie_sprzedawcy,
SPRZEDAWCY.nazwisko AS nazwisko_sprzedawcy,
COUNT(*) AS liczba_sprzedanych_aut
FROM ZAMOWIENIA
INNER JOIN SAMOCHODY on zamowienia.id_samochodu = samochody.id_samochodu
INNER JOIN SPRZEDAWCY on ZAMOWIENIA.id_sprzedawcy = SPRZEDAWCY.id_sprzedawcy
GROUP BY SPRZEDAWCY.id_sprzedawcy, SPRZEDAWCY.imie, SPRZEDAWCY.nazwisko;
[================================================================================
====================]
```

4.3.

```
[================================================================================]
[=============PROCEDURA TWORZACA PODSUMOWANIE ================================]
[=============SPRZEDANYCH AUT PRZEZ WSKAZANEGO PRACOWNIKA=================]
[=============WE WSKAZANYM OKRESIE=========================================]
[================================================================================]
DROP TABLE PODSUMOWANIE4;
CREATE TABLE PODSUMOWANIE4
(id_sprzedawcy NUMBER(10) PRIMARY KEY,
imie_sprzedawcy VARCHAR(20) NOT NULL,
nazwisko_sprzedawcy VARCHAR(30) NOT NULL,
liczba_sprzedanych_aut NUMBER(10))

CREATE OR REPLACE PROCEDURE generuj_podsumowanie (
p_id_sprzedawcy ZAMOWIENIA.id_sprzedawcy%TYPE,
p_data_zamowienia_od ZAMOWIENIA.data_zamowienia%TYPE,
p_data_zamowienia_do ZAMOWIENIA.data_zamowienia%TYPE

)
IS
    v_id_sprzedawcy SPRZEDAWCY.id_sprzedawcy%TYPE;
    v_imie SPRZEDAWCY.imie%TYPE;
    v_nazwisko SPRZEDAWCY.nazwisko%TYPE;
    v_liczba_sprzedanych_aut NUMBER(10);
BEGIN
```

```
SELECT
SPRZEDAWCY.id_sprzedawcy,
SPRZEDAWCY.imie AS imie_sprzedawcy,
SPRZEDAWCY.nazwisko AS nazwisko_sprzedawcy,
COUNT(*) AS liczba_sprzedanych_aut INTO v_id_sprzedawcy, v_imie, v_nazwisko,
v_liczba_sprzedanych_aut
FROM ZAMOWIENIA
INNER JOIN SAMOCHODY on zamowienia.id_samochodu = samochody.id_samochodu
INNER JOIN SPRZEDAWCY on ZAMOWIENIA.id_sprzedawcy = SPRZEDAWCY.id_sprzedawcy
WHERE data_zamowienia BETWEEN p_data_zamowienia_od AND p_data_zamowienia_do
AND SPRZEDAWCY.id_sprzedawcy=p_id_sprzedawcy
GROUP BY SPRZEDAWCY.id_sprzedawcy, SPRZEDAWCY.imie, SPRZEDAWCY.nazwisko;

INSERT INTO PODSUMOWANIE4(id_sprzedawcy, imie_sprzedawcy, nazwisko_sprzedawcy,
liczba_sprzedanych_aut)
VALUES (v_id_sprzedawcy, v_imie, v_nazwisko, v_liczba_sprzedanych_aut);
END;

DECLARE
BEGIN
DELETE PODSUMOWANIE4;
generuj_podsumowanie(1, '01/04/21', '30/04/21'); -- id pracownika, data_od,
data_do
END;

[========WYZWALACZ INSERT PODSUMOWANIE4========]
[==============================================]
set serveroutput on;
CREATE OR REPLACE TRIGGER insert_podsumowanie4
after insert
on podsumowanie4
begin
dbms_output.put_line('[Wyzwalacz] Wygenerowano raport!');
end;
[==============================================]
[==============================================]


[=============================================================================]
[=============================================================================]
[=============================================================================]
4.4.
[======= PROCEDURA TWORZACA PODSUMOWANIE SPRZEDANYCH AUT ====================]
[================ PRZEZ WSZYSTKICH PRACOWNIKÓW ==============================]
[=============================================================================]
DROP TABLE PODSUMOWANIE5;
CREATE TABLE PODSUMOWANIE5
(id_sprzedawcy NUMBER(10) PRIMARY KEY,
imie_sprzedawcy VARCHAR(20) NOT NULL,
nazwisko_sprzedawcy VARCHAR(30) NOT NULL,
liczba_sprzedanych_aut NUMBER(10))
CREATE OR REPLACE PROCEDURE generuj_podsumowanie2
IS
v_ile NUMBER(10):=0;
v_i NUMBER(10):=0;
v_id_sprzedawcy SPRZEDAWCY.id_sprzedawcy%TYPE;
v_imie_sprzedawcy SPRZEDAWCY.imie%TYPE;
```

```
v_nazwisko_sprzedawcy SPRZEDAWCY.nazwisko%TYPE;
v_liczba_sprzedanych_aut NUMBER;

BEGIN

    SELECT COUNT(*) INTO v_ile FROM (SELECT
    SPRZEDAWCY.id_sprzedawcy,
    SPRZEDAWCY.imie AS imie_sprzedawcy,
    SPRZEDAWCY.nazwisko AS nazwisko_sprzedawcy,
    COUNT(*) AS liczba_sprzedanych_aut
    FROM ZAMOWIENIA
    INNER JOIN SAMOCHODY on zamowienia.id_samochodu = samochody.id_samochodu
    INNER JOIN SPRZEDAWCY on ZAMOWIENIA.id_sprzedawcy =
SPRZEDAWCY.id_sprzedawcy
    GROUP BY SPRZEDAWCY.id_sprzedawcy, SPRZEDAWCY.imie, SPRZEDAWCY.nazwisko);

    LOOP
        --SELECT department_id INTO tab(v_i) FROM (SELECT department_id,
ROWNUM as RN FROM DEPARTMENTS) WHERE RN = (v_i+1);

        SELECT id_sprzedawcy, imie_sprzedawcy, nazwisko_sprzedawcy,
liczba_sprzedanych_aut
        INTO v_id_sprzedawcy, v_imie_sprzedawcy, v_nazwisko_sprzedawcy,
v_liczba_sprzedanych_aut
        FROM (SELECT
        SPRZEDAWCY.id_sprzedawcy,
        SPRZEDAWCY.imie AS imie_sprzedawcy,
        SPRZEDAWCY.nazwisko AS nazwisko_sprzedawcy,
        COUNT(*) AS liczba_sprzedanych_aut
        FROM ZAMOWIENIA
        INNER JOIN SAMOCHODY on zamowienia.id_samochodu =
samochody.id_samochodu
        INNER JOIN SPRZEDAWCY on ZAMOWIENIA.id_sprzedawcy =
SPRZEDAWCY.id_sprzedawcy
        GROUP BY SPRZEDAWCY.id_sprzedawcy, SPRZEDAWCY.imie,
SPRZEDAWCY.nazwisko)
        OFFSET v_i ROWS FETCH NEXT 1 ROWS ONLY;

        --DBMS_OUTPUT.put_line(v_id_sprzedawcy||' '||v_imie_sprzedawcy||'
'||v_nazwisko_sprzedawcy||' '||v_liczba_sprzedanych_aut);

        INSERT INTO PODSUMOWANIE5(id_sprzedawcy, imie_sprzedawcy,
nazwisko_sprzedawcy, liczba_sprzedanych_aut)
        VALUES (v_id_sprzedawcy, v_imie_sprzedawcy, v_nazwisko_sprzedawcy,
v_liczba_sprzedanych_aut);
        v_i:=v_i+1;
        EXIT WHEN v_i>=v_ile;
    END LOOP;
END;

DECLARE
BEGIN
DELETE PODSUMOWANIE5;
generuj_podsumowanie2;
END;
[================================================================================]
[================================================================================]
```

5. ***Całość – wszystkie komendy wrzucone do repozytorium.*** ***[DONE]***

Dodatkowo:
Testy przygotowanych procedur/funkcji/triggerów itp.

[===]
[===== DODATKOWO DODAŁEM TESTY =========]
[===]
SET SERVEROUTPUT ON;

1. Procedura dodająca nowego sprzedawcę
DECLARE
BEGIN
dodaj_sprzedawce(4, 1, 'Nowy', 'Jan', '', 'Handlarz', '05/04/21', 6000, 999222999);
END;

2. Procedura aktualizująca pensje sprzedawcy
DECLARE
BEGIN
aktualizuj_pensje_sprzedawcy(4, 7000);
END;

3. Funkcja generujaca adres e-mail wskazanego sprzedawcy
DECLARE
v_mail VARCHAR2(100);
BEGIN
v_mail := wygenerujmail(4);
dbms_output.PUT_LINE(v_mail);
END;

4. PROCEDURA AKTUALIZUJACA ADRES EMAIL WSKAZANEMU PRACOWNIKOWI - WYKORZYSTUJACA FUNKCJE GENEROWANIA MAILA
DECLARE
BEGIN
ZAKTUALIZUJ_MAILA_SPRZEDAWCY(4);
END;

5. Procedura usuwająca sprzedawce
DECLARE
BEGIN
usun_sprzedawce(4);
END;

6. FUNKCJA SPRAWDZAJACA CZY WPROWADZONY VIN SAMOCHODU JEST POPRAWNY
DECLARE

```
v_czy_poprawny NUMBER(1,0);
BEGIN
v_czy_poprawny := SPRAWDZ_POPRAWNOSC_VIN('1234567891ASDFGHJ');
dbms_output.PUT_LINE(v_czy_poprawny);
if v_czy_poprawny = 1 THEN
dbms_output.PUT_LINE('poprawny nr vin');
else
dbms_output.PUT_LINE('NIEPOPRAWNY nr vin');
end if;
END;
```

7. PROCEDURA DODAJACA NOWY SAMOCHOD -- WYKORZYSTUJACA FUNKCJE SPRAWDZANIA NR VIN

```
-- poprawny vin
DECLARE
BEGIN
dodaj_samochod(2, 'Volkswagen', 'Golf VIII', 2020, 1968, 'Diesel', 115, 1, 'WV2WZ2ZRZBY294825',
102900, 0);
END;

-- niepoprawny vin
DECLARE
BEGIN
dodaj_samochod(2, 'Volkswagen', 'Golf VIII', 2020, 1968, 'Diesel', 115, 1, 'WV2WZ2ZRZBY29482', 102900,
0);
END;
```

8. FUNKCJA SPRAWDZAJACA CZY WPROWADZONY NR TELEFONU KLIENTA JEST POPRAWNY

```
DECLARE
v_czy_poprawny NUMBER(1,0);
BEGIN
v_czy_poprawny := SPRAWDZ_POPRAWNOSC_NR_TELEFONU('123456789');
dbms_output.PUT_LINE(v_czy_poprawny);
if v_czy_poprawny = 1 THEN
dbms_output.PUT_LINE('poprawny nr telefonu');
else
dbms_output.PUT_LINE('NIEPOPRAWNY nr telefonu');
end if;
END;
```

9. PROCEDURA DODAJACA NOWEGO KLIENTA -- WYKORZYSTUJACA FUNKCJE SPRAWDZANIA NR TELEFONU

```
-- poprawny nr telefonu
DECLARE
```

```
BEGIN
dodaj_klienta('Wysoki', 'Jan', 'Warszawa', 'Wiejska 1', '00-500', 123456789);
END;

-- niepoprawny nr telefonu
DECLARE
BEGIN
dodaj_klienta('Wysoki', 'Jan', 'Warszawa', 'Wiejska 1', '00-500', 123);
END;
```

10. TRIGGERY (WYZWALACZE) Przejdzmy do tabeli samochody_historia i skasujmy samochod o id_samochodu = 11

```
delete samochody where id_samochodu = 11;
```

Zostanie uruchomiony wyzwalacz delete_samochody W tabeli samochody_historia powinien się pojawić usunięty samochód.

11. Polecenie update na tabeli samochody również wywoła odpowiedni trigger update_samochody i zapisze odpowiednie logi w tabeli samochody_logi

```
update samochody set model_auta='Altea2' where model_auta='Altea'
```

12. Podobnie jak w pkt 11 działa opcja wstawiania rekordu do tabeli samochody - również uruchamiany jest odpowiedni trigger i zapisywane sa logi w tabeli samochody_logi

```
INSERT INTO Samochody
(
id_samochodu, id_komisu, marka, model_auta, rok_produkcji, pojemnosc_cm3, rodzaj_paliwa, moc_km,
przebieg, nr_vin, kwota_do_zaplaty, czy_sprzedany
)
VALUES (18, 3, 'Seat', 'Altea', 2009, 1790, 'Benzyna', 160, 1420005, 'VSSZZZSPZ9R002604', 157999, 0)
```

13. Stworzono widok - zestawienie1 = ZESTAWIENIE TWORZACE PODSUMOWANIE SPRZEDANYCH AUT PRZEZ PRACOWNIKA NR 1 W MIESIACU KWIETNIU
oraz widok zestawienie1 = ZESTAWIENIE TWORZACE PODSUMOWANIE SPRZEDANYCH AUT PRZEZ WSZYSTKICH PRACOWNIKÓW

14. PROCEDURA TWORZACA PODSUMOWANIE SPRZEDANYCH AUT PRZEZ WSKAZANEGO PRACOWNIKA

```
DECLARE
BEGIN
DELETE PODSUMOWANIE4;
generuj_podsumowanie(1, '01/04/21', '30/04/21'); -- id pracownika, data_od, data_do
END;
```

W tabeli podsumowanie4 zostanie wygenerowane podsumowanie sprzedanych aut przez wskazanego pracownika
Dodatkowo zostanie wywolany wyzwalacz insert_podsumowanie4 i na ekranie powinien pojawic sie komunikat:
"[Wyzwalacz] Wygenerowano raport!"

15. PROCEDURA TWORZACA PODSUMOWANIE SPRZEDANYCH AUT PRZEZ WSZYSTKICH PRACOWNIKÓW

```
DECLARE
BEGIN
DELETE PODSUMOWANIE5;
generuj_podsumowanie2;
END;
```

W tabeli podsumowanie5 zostanie wygenerowanie podsumowanie aut sprzedanych przez wskazanych pracowników