Uniwersytet Gdański

SERWERY BAZY DANYCH

Projekt: Komis samochodowy
Środowisko: Oracle
Zadani: Zadanie 8 (Oracle - typy obiektowe)

Prowadzący:
dr Robert Fidytek

Wykonała:
Paulina Kimak
Nr albumu 292511

Gdańsk, 2026

Spis treści

1. Opis Projektu

Zaprojektowano komis samochodowy, w którym przechowywane są oferty aut oraz dane klientów zainteresowanych zakupem. Komis rejestruje zdarzenia związane z kontaktem klienta z konkretnym autem, w szczególności jazdy próbne oraz rezerwacje oglądania.

- **Encja A (Samochód)** opisuje pojazd wystawiony w komisie wraz z cena i danymi identyfikacyjnymi.
- **Encja B (Klient)** przechowuje dane kontaktowe osoby zainteresowanej.
- **Encja zdarzenia (Zdarzenie)** łączy klienta z autem i przechowuje informacje o typie zdarzenia oraz czasie jego wystąpienia.

Relacje:

- jeden klient może mieć wiele zdarzeń,
- jedno auto może mieć wiele zdarzeń;
- każde zdarzenie dotyczy dokładnie jednego auta i jednego klienta.

Zdarzenie jest bytem, który wprost realizuje powiązanie pomiędzy dwoma bytami: Samochód (A) i Klient (B).

2. Typy obiektowe

Zdefiniowano trzy typy obiektowe:
• **T_CAR** - encja A (samochód), atrybuty: car_id, vin, marka, model, rok_prod, cena_netto, waluta wraz metodami

```
29  CREATE OR REPLACE TYPE t_car AS OBJECT (
30    car_id      NUMBER(10),
31    vin         VARCHAR2(20),
32    marka       VARCHAR2(30),
33    model       VARCHAR2(30),
34    rok_prod    NUMBER(4),
35    cena_netto  NUMBER(10,2),
36    waluta      VARCHAR2(3),
37
38    -- (6a) Metoda obliczeniowa
39    MEMBER FUNCTION cena_brutto(p_vat NUMBER DEFAULT 0.23) RETURN NUMBER,
40
41    -- (6b) Metoda opisowa
42    MEMBER FUNCTION opis RETURN VARCHAR2,
43
44    -- (8A) MAP METHOD - sortowanie po cenie netto
45    MAP MEMBER FUNCTION map_key RETURN NUMBER
46  );
47  /
48
49  CREATE OR REPLACE TYPE BODY t_car AS
50
51    MEMBER FUNCTION cena_brutto(p_vat NUMBER DEFAULT 0.23) RETURN NUMBER IS
52    BEGIN
53      RETURN ROUND(self.cena_netto * (1 + NVL(p_vat,0)), 2);
54    END;
55
56    MEMBER FUNCTION opis RETURN VARCHAR2 IS
57    BEGIN
58      RETURN self.marka || ' ' || self.model ||
59             ' (' || self.rok_prod || '), VIN=' || self.vin;
60    END;
61
62    MAP MEMBER FUNCTION map_key RETURN NUMBER IS
63    BEGIN
64      RETURN NVL(self.cena_netto, 0);
65    END;
66
67  END;
68  /
69
```

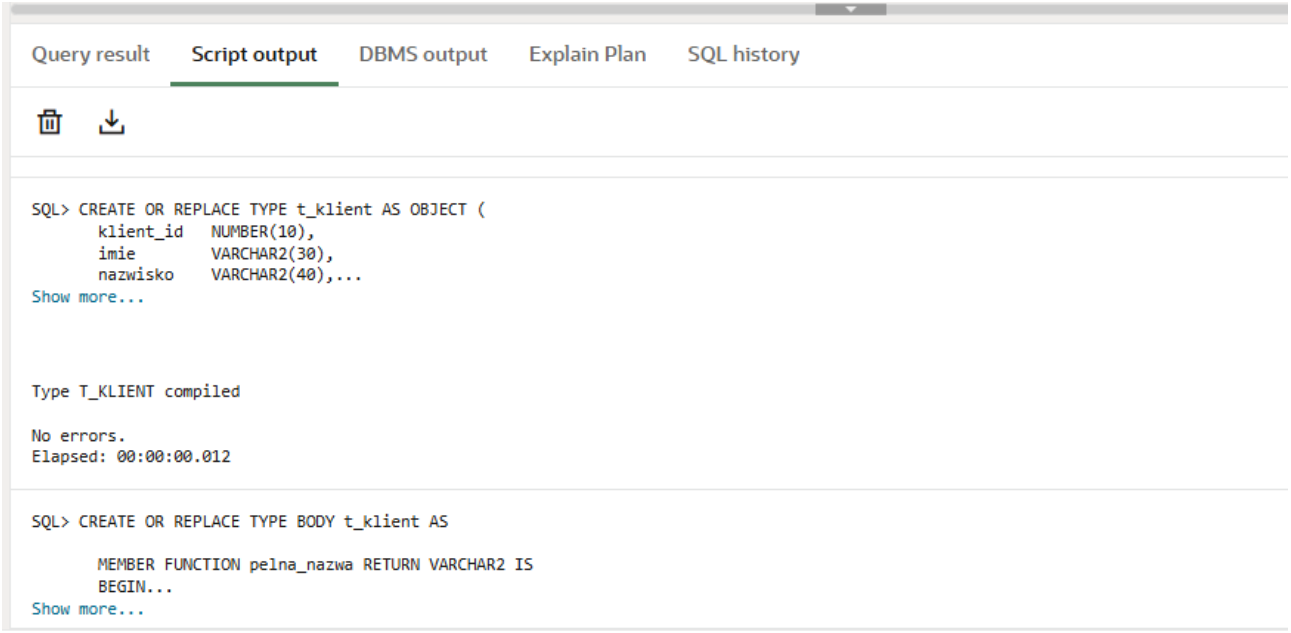
f

Query result | Script output | DBMS output | Explain Plan | SQL history

```
SQL> PROMPT === [01] Typy obiektowe ===


=== [01] Typy obiektowe ===


SQL> CREATE OR REPLACE TYPE t_car AS OBJECT (
       car_id      NUMBER(10),
       vin         VARCHAR2(20),
       marka       VARCHAR2(30),...
Show more...



Type T_CAR compiled

No errors.
Elapsed: 00:00:00.031
```

• **T_KLIENT** - encja B (klient), atrybuty: klient_id, imie, nazwisko, email, data_ur.

```
72  ------------------------------------------------------------------------
73  CREATE OR REPLACE TYPE t_klient AS OBJECT (
74    klient_id   NUMBER(10),
75    imie        VARCHAR2(30),
76    nazwisko    VARCHAR2(40),
77    email       VARCHAR2(80),
78    data_ur     DATE,
79
80    MEMBER FUNCTION pelna_nazwa RETURN VARCHAR2
81  );
82  /
83
84  CREATE OR REPLACE TYPE BODY t_klient AS
85
86    MEMBER FUNCTION pelna_nazwa RETURN VARCHAR2 IS
87    BEGIN
88      RETURN self.imie || ' ' || self.nazwisko;
89    END;
90
91  END;
92  /
93
```



• **T_EVT** - encja zdarzenia, atrybuty: evt_id, evt_ts, evt_typ, czas_min, notatka oraz 2 referencje REF: car_ref (REF T_CAR) i klient_ref (REF T_KLIENT).

```
CREATE OR REPLACE TYPE t_evt AS OBJECT (
  evt_id      NUMBER(10),
  evt_ts      TIMESTAMP,
  evt_typ     VARCHAR2(20),
  car_ref     REF t_car,
  klient_ref  REF t_klient,
  czas_min    NUMBER(5),
  notatka     VARCHAR2(200),

  MEMBER FUNCTION koszt RETURN NUMBER,
  MEMBER FUNCTION to_text RETURN VARCHAR2
);
/

CREATE OR REPLACE TYPE BODY t_evt AS

  MEMBER FUNCTION koszt RETURN NUMBER IS
  BEGIN
    IF UPPER(self.evt_typ) = 'JAZDA' THEN
      RETURN NVL(self.czas_min,0) * 5;
    ELSE
      RETURN 0;
    END IF;
  END;

  MEMBER FUNCTION to_text RETURN VARCHAR2 IS
  BEGIN
    RETURN 'EVT['||self.evt_id||'] '||self.evt_typ||' @ '||
           TO_CHAR(self.evt_ts,'YYYY-MM-DD HH24:MI')||
           ', min='||NVL(TO_CHAR(self.czas_min),'NULL')||
           ', note='||NVL(self.notatka,'-');
  END;

END;
/
```

Query result | Script output | DBMS output | Explain Plan | SQL history

```
Elapsed: 00:00:00.008

SQL> CREATE OR REPLACE TYPE t_evt AS OBJECT (
       evt_id      NUMBER(10),
       evt_ts      TIMESTAMP,
       evt_typ     VARCHAR2(20),...
Show more...



Type T_EVT compiled

No errors.
Elapsed: 00:00:00.010

SQL> CREATE OR REPLACE TYPE BODY t_evt AS

       MEMBER FUNCTION koszt RETURN NUMBER IS
       BEGIN...
```

3. Tabele obiektowe oraz dane

Utworzono tabele obiektowe:

```
CREATE TABLE car_tab OF t_car (
  CONSTRAINT car_pk PRIMARY KEY (car_id),
  CONSTRAINT car_vin_uq UNIQUE (vin)
);

CREATE TABLE klient_tab OF t_klient (
  CONSTRAINT klient_pk PRIMARY KEY (klient_id),
  CONSTRAINT klient_email_uq UNIQUE (email)
);

CREATE TABLE evt_tab OF t_evt (
  CONSTRAINT evt_pk PRIMARY KEY (evt_id)
);
```

Inserty

Wstawiono co najmniej 10 rekordów do każdej z tabel:

Car insert

[SQL Worksheet]*

```
SET SERVEROUTPUT ON

PROMPT === [02] INSERT: CAR_TAB (A) ===
INSERT INTO car_tab VALUES (t_car(1,'VIN00000000000001','Toyota','Corolla',2017, 42000,'PLN'));
INSERT INTO car_tab VALUES (t_car(2,'VIN00000000000002','Toyota','Yaris',2019, 52000,'PLN'));
INSERT INTO car_tab VALUES (t_car(3,'VIN00000000000003','Skoda','Octavia',2016, 39000,'PLN'));
INSERT INTO car_tab VALUES (t_car(4,'VIN00000000000004','Volkswagen','Golf',2018, 61000,'PLN'));
INSERT INTO car_tab VALUES (t_car(5,'VIN00000000000005','Ford','Focus',2015, 31000,'PLN'));
INSERT INTO car_tab VALUES (t_car(6,'VIN00000000000006','BMW','320i',2014, 57000,'PLN'));
INSERT INTO car_tab VALUES (t_car(7,'VIN00000000000007','Audi','A4',2013, 54000,'PLN'));
INSERT INTO car_tab VALUES (t_car(8,'VIN00000000000008','Hyundai','i30',2020, 69000,'PLN'));
INSERT INTO car_tab VALUES (t_car(9,'VIN00000000000009','Kia','Ceed',2021, 75000,'PLN'));
INSERT INTO car_tab VALUES (t_car(10,'VIN00000000000010','Mazda','3',2018, 63000,'PLN'));
COMMIT;
```

```
SQL> PROMPT === [02] INSERT: CAR_TAB (A) ===

=== [02] INSERT: CAR_TAB (A) ===

SQL> INSERT INTO car_tab VALUES (t_car(1,'VIN00000000000001','Toyota','Corolla',2017, 42000,'PLN'))

1 row inserted.

Elapsed: 00:00:00.158

SQL> INSERT INTO car_tab VALUES (t_car(2,'VIN00000000000002','Toyota','Yaris',2019, 52000,'PLN'))
```

Client insert

```
PROMPT === [02] INSERT: KLIENT_TAB (B) ===
INSERT INTO klient_tab VALUES (t_klient(1,'Anna','Nowak','anna.nowak@example.com', DATE '1994-03-12'));
INSERT INTO klient_tab VALUES (t_klient(2,'Piotr','Kowalski','piotr.kowalski@example.com', DATE '1988-11-02'));
INSERT INTO klient_tab VALUES (t_klient(3,'Katarzyna','Wisniewska','k.wisniewska@example.com', DATE '1990-07-21'));
INSERT INTO klient_tab VALUES (t_klient(4,'Marek','Wojcik','marek.wojcik@example.com', DATE '1985-01-09'));
INSERT INTO klient_tab VALUES (t_klient(5,'Ewa','Kaczmarek','ewa.k@example.com', DATE '1997-05-30'));
INSERT INTO klient_tab VALUES (t_klient(6,'Tomasz','Mazur','t.mazur@example.com', DATE '1992-12-14'));
INSERT INTO klient_tab VALUES (t_klient(7,'Olga','Zielinska','olga.z@example.com', DATE '1999-09-03'));
INSERT INTO klient_tab VALUES (t_klient(8,'Pawel','Szymanski','pawel.s@example.com', DATE '1983-06-18'));
INSERT INTO klient_tab VALUES (t_klient(9,'Agnieszka','Dabrowska','aga.d@example.com', DATE '1991-02-26'));
INSERT INTO klient_tab VALUES (t_klient(10,'Michal','Lewandowski','m.lewandowski@example.com', DATE '1989-10-05'));
COMMIT;
```

Query result | Script output | DBMS output | Explain Plan | SQL history

```
SQL> PROMPT === [02] INSERT: KLIENT_TAB (B) ===

=== [02] INSERT: KLIENT_TAB (B) ===

SQL> INSERT INTO klient_tab VALUES (t_klient(1,'Anna','Nowak','anna.nowak@example.com', DATE '1994-03-12'))

1 row inserted.

Elapsed: 00:00:00.133

SQL> INSERT INTO klient_tab VALUES (t_klient(2,'Piotr','Kowalski','piotr.kowalski@example.com', DATE '1988-11-02'))

1 row inserted.

Elapsed: 00:00:00.002
```

4. REF/DEREF + SCOPE

W typie T_EVT zastosowano dwa pola REF: **car_ref** oraz **klient_ref.**
Utworzono tabele EVT_TAB OF T_EVT, a następnie ustawiono ograniczenia zakresu referencji (SCOPE IS):
• ALTER TABLE evt_tab ADD SCOPE FOR (car_ref) IS car_tab;
• ALTER TABLE evt_tab ADD SCOPE FOR (klient_ref) IS klient_tab;

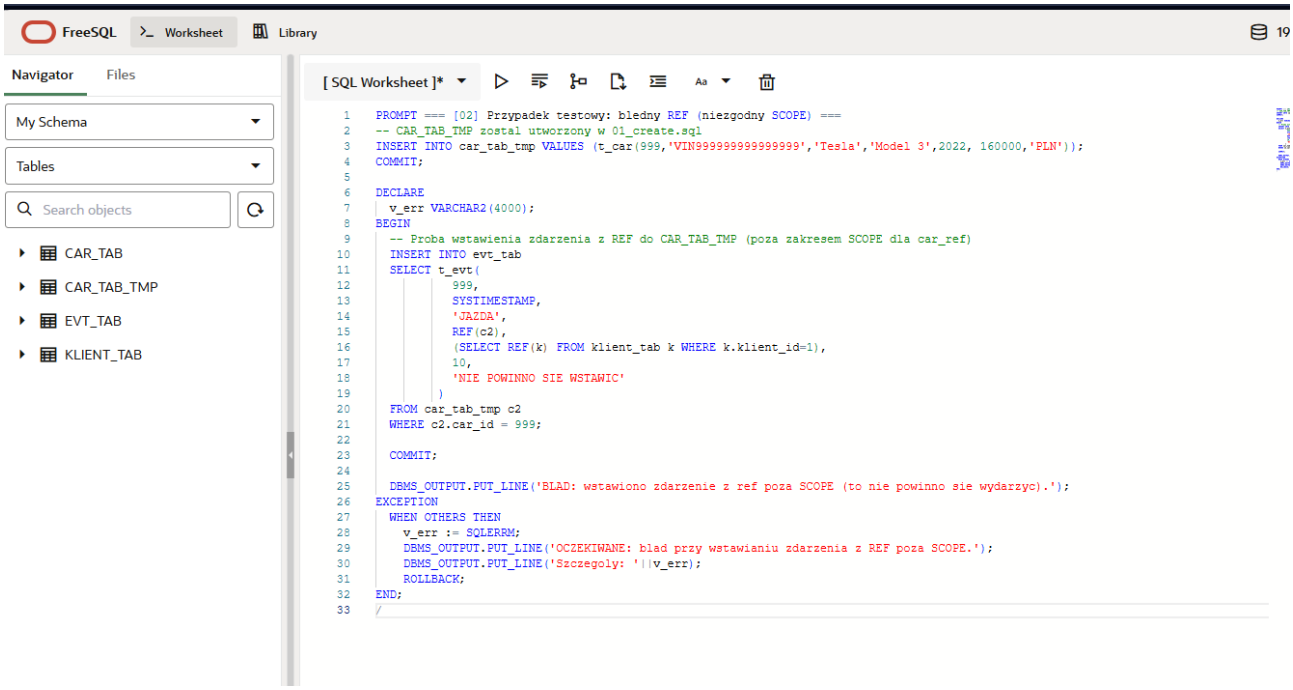
```
[ SQL Worksheet ]*
1   ------------------------------------------------------------------------
2   -- Tabela pomocnicza do testu REF poza SCOPE
3   ------------------------------------------------------------------------
4   CREATE TABLE car_tab_tmp OF t_car;
5
6   ------------------------------------------------------------------------
7   -- (4) SCOPE dla referencji
8   ------------------------------------------------------------------------
9   ALTER TABLE evt_tab ADD SCOPE FOR (car_ref) IS car_tab;
10  ALTER TABLE evt_tab ADD SCOPE FOR (klient_ref) IS klient_tab;
```

Wstawiono 15 rekordów do EVT_TAB

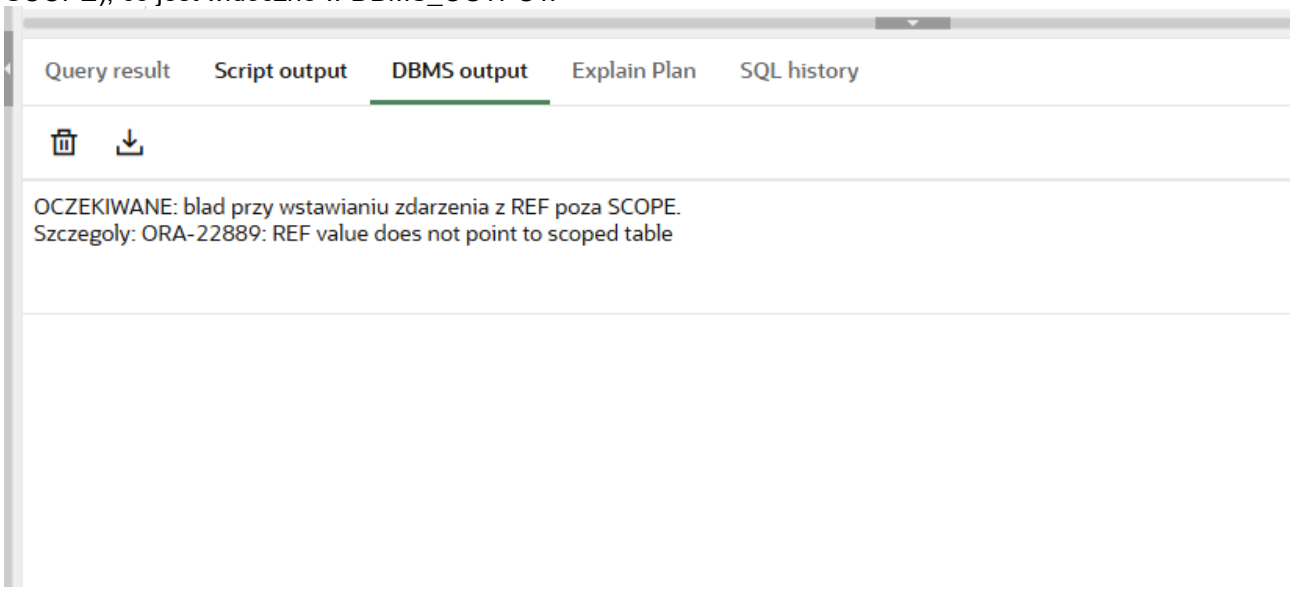
```
[ SQL Worksheet ]*
33  INSERT INTO evt_tab
34  SELECT t_evt(101, SYSTIMESTAMP-INTERVAL '10' DAY, 'JAZDA', REF(c), REF(k), 30, 'pierwsza jazda')
35  FROM car_tab c, klient_tab k WHERE c.car_id=1 AND k.klient_id=1;
36
37  INSERT INTO evt_tab
38  SELECT t_evt(102, SYSTIMESTAMP-INTERVAL '9' DAY, 'REZERW', REF(c), REF(k), NULL, 'rezerwacja na ogladanie')
39  FROM car_tab c, klient_tab k WHERE c.car_id=2 AND k.klient_id=2;
40
41  INSERT INTO evt_tab
42  SELECT t_evt(103, SYSTIMESTAMP-INTERVAL '9' DAY, 'JAZDA', REF(c), REF(k), 20, 'miasto')
43  FROM car_tab c, klient_tab k WHERE c.car_id=2 AND k.klient_id=3;
44
45  INSERT INTO evt_tab
46  SELECT t_evt(104, SYSTIMESTAMP-INTERVAL '8' DAY, 'JAZDA', REF(c), REF(k), 45, 'trasa S8')
47  FROM car_tab c, klient_tab k WHERE c.car_id=3 AND k.klient_id=4;
48
49  INSERT INTO evt_tab
50  SELECT t_evt(105, SYSTIMESTAMP-INTERVAL '8' DAY, 'REZERW', REF(c), REF(k), NULL, 'zaliczka potwierdzona')
51  FROM car_tab c, klient_tab k WHERE c.car_id=4 AND k.klient_id=5;
52
53  INSERT INTO evt_tab
54  SELECT t_evt(106, SYSTIMESTAMP-INTERVAL '7' DAY, 'JAZDA', REF(c), REF(k), 15, 'parking')
55  FROM car_tab c, klient_tab k WHERE c.car_id=4 AND k.klient_id=1;
56
57  INSERT INTO evt_tab
58  SELECT t_evt(107, SYSTIMESTAMP-INTERVAL '7' DAY, 'JAZDA', REF(c), REF(k), 35, 'obwodnica')
59  FROM car_tab c, klient_tab k WHERE c.car_id=5 AND k.klient_id=6;
60
61  INSERT INTO evt_tab
62  SELECT t_evt(108, SYSTIMESTAMP-INTERVAL '6' DAY, 'REZERW', REF(c), REF(k), NULL, 'czeka na finansowanie')
63  FROM car_tab c, klient_tab k WHERE c.car_id=6 AND k.klient_id=7;
64
65  INSERT INTO evt_tab
66  SELECT t_evt(109, SYSTIMESTAMP-INTERVAL '6' DAY, 'JAZDA', REF(c), REF(k), 25, 'krotka jazda')
67  FROM car_tab c, klient_tab k WHERE c.car_id=6 AND k.klient_id=8;
68
69  INSERT INTO evt_tab
70  SELECT t_evt(110, SYSTIMESTAMP-INTERVAL '5' DAY, 'JAZDA', REF(c), REF(k), 40, 'autostrada')
71  FROM car_tab c, klient_tab k WHERE c.car_id=7 AND k.klient_id=9;
72
73  INSERT INTO evt_tab
74  SELECT t_evt(111, SYSTIMESTAMP-INTERVAL '5' DAY, 'REZERW', REF(c), REF(k), NULL, 'rezerwacja weekend')
75  FROM car_tab c, klient_tab k WHERE c.car_id=8 AND k.klient_id=10;
76
77  INSERT INTO evt_tab
78  SELECT t_evt(112, SYSTIMESTAMP-INTERVAL '4' DAY, 'JAZDA', REF(c), REF(k), 50, 'trasa podmiejska')
79  FROM car_tab c, klient_tab k WHERE c.car_id=8 AND k.klient_id=2;
```

```
Query result | Script output | DBMS output | Explain Plan | SQL history

SQL> PROMPT === [02] INSERT: EVT_TAB (zdarzenia) ===

=== [02] INSERT: EVT_TAB (zdarzenia) ===

SQL> INSERT INTO evt_tab
     SELECT t_evt(101, SYSTIMESTAMP-INTERVAL '10' DAY, 'JAZDA', REF(c), REF(k), 30, 'pierwsza jazda')
     FROM car_tab c, klient_tab k WHERE c.car_id=1 AND k.klient_id=1

1 row inserted.

Elapsed: 00:00:00.088

SQL> INSERT INTO evt_tab
     SELECT t_evt(102, SYSTIMESTAMP-INTERVAL '9' DAY, 'REZERW', REF(c), REF(k), NULL, 'rezerwacja na ogladanie')
     FROM car_tab c, klient_tab k WHERE c.car_id=2 AND k.klient_id=2
```

Przypadek testowy
Dodatkowo wykonano przypadek testowy, który próbuje wstawić zdarzenie z REF wskazującym na obiekt auta w innej tabeli (CAR_TAB_TMP).



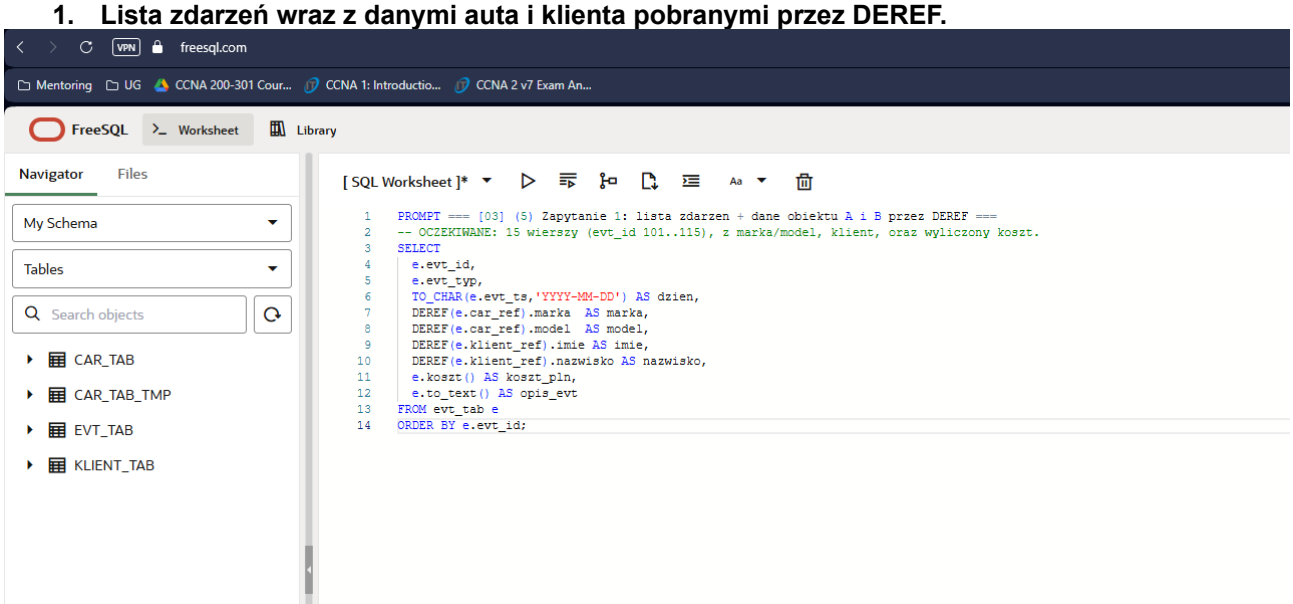
Z powodu ustawionego SCOPE, taki INSERT zostaje odrzucony przez Oracle (błąd wstawiania REF poza SCOPE), co jest widoczne w DBMS_OUTPUT.



5. Zapytania z VALUE/DEREF

W pliku 03_tests.sql znajdują sie 3 zapytania, które wykorzystują VALUE i/lub DEREF:

1. **Lista zdarzeń wraz z danymi auta i klienta pobranymi przez DEREF.**



Query result | Script output | DBMS output | Explain Plan | SQL history

Download | Execution time: 0.037 seconds

	EVT_ID	EVT_TYP	DZIEN	MARKA	MODEL	IMIE	NAZWISKO	KOSZT_
1	101	JAZDA	2026-01-06	Toyota	Corolla	Anna	Nowak	
2	102	REZERW	2026-01-07	Toyota	Yaris	Piotr	Kowalski	
3	103	JAZDA	2026-01-07	Toyota	Yaris	Katarzyna	Wisniewska	
4	104	JAZDA	2026-01-08	Skoda	Octavia	Marek	Wojcik	
5	105	REZERW	2026-01-08	Volkswagen	Golf	Ewa	Kaczmarek	
6	106	JAZDA	2026-01-09	Volkswagen	Golf	Anna	Nowak	
7	107	JAZDA	2026-01-09	Ford	Focus	Tomasz	Mazur	
8	108	REZERW	2026-01-10	BMW	320i	Olga	Zielinska	
9	109	JAZDA	2026-01-10	BMW	320i	Pawel	Szymanski	
10	110	JAZDA	2026-01-11	Audi	A4	Agnieszka	Dabrowska	
11	111	REZERW	2026-01-11	Hyundai	i30	Michal	Lewandowski	
12	112	JAZDA	2026-01-12	Hyundai	i30	Piotr	Kowalski	

2. Raport zagregowany: liczba zdarzeń i suma kosztów na auto (GROUP BY po REF).

FreeSQL | Worksheet | Library

Navigator | Files

My Schema

Tables

Search objects

- CAR_TAB
- CAR_TAB_TMP
- EVT_TAB
- KLIENT_TAB

[SQL Worksheet]*

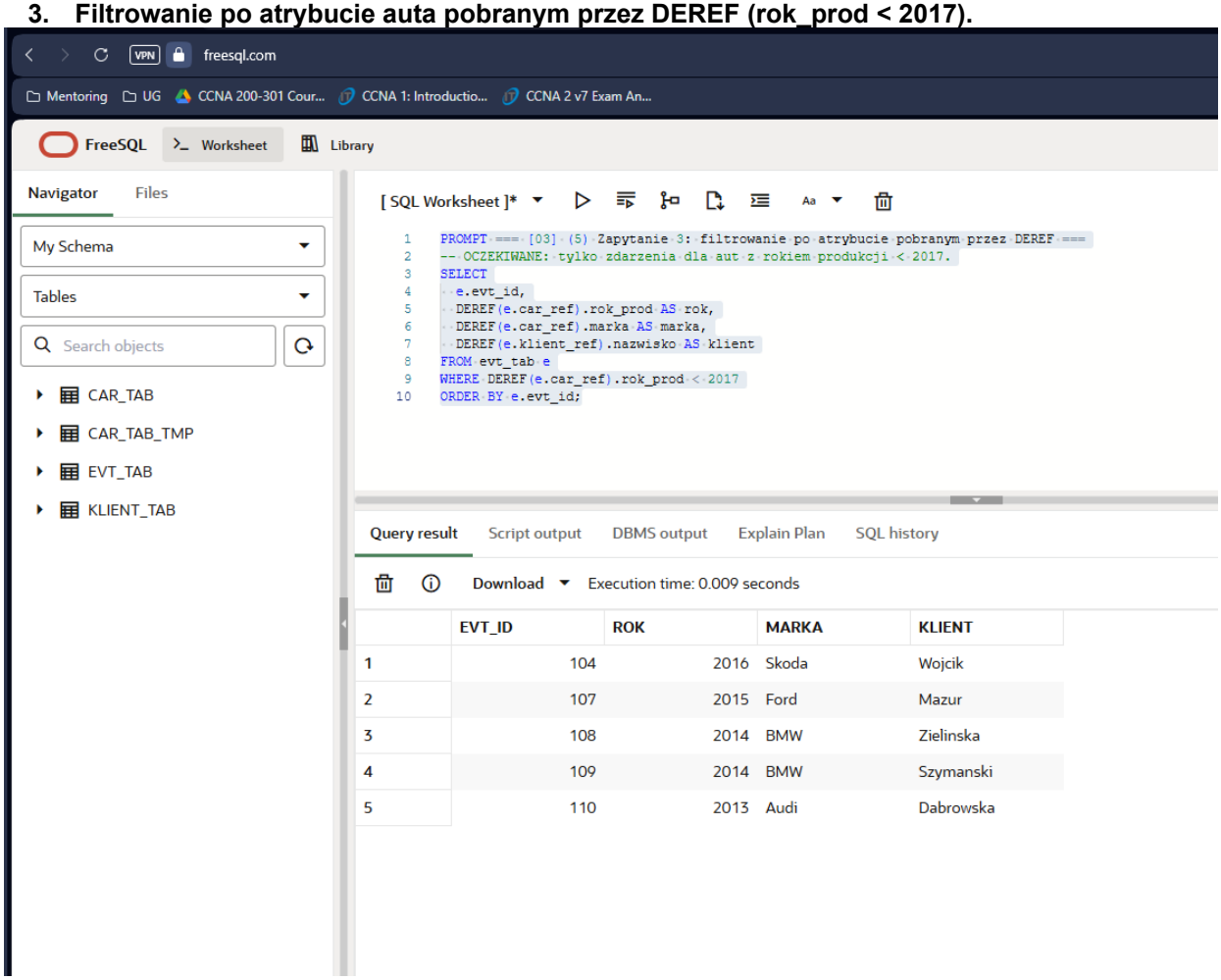
```
1
2  PROMPT === [03] (5) Zapytanie 2: agregacja - liczba zdarzen na samochod ===
3  -- OCZEKIWANE: do 10 wierszy; pokazuje liczbę zdarzeń i sumę kosztów na VIN
4  SELECT
5    DEREF(e.car_ref).vin AS vin,
6    DEREF(e.car_ref).marka || ' ' || DEREF(e.car_ref).model AS auto,
7    COUNT(*) AS liczba_zdarzen,
8    SUM(e.koszt()) AS suma_kosztow_pln
9  FROM evt_tab e
10 GROUP BY
11   DEREF(e.car_ref).vin,
12   DEREF(e.car_ref).marka,
13   DEREF(e.car_ref).model
14 ORDER BY liczba_zdarzen DESC, suma_kosztow_pln DESC;
15
```

Query result | Script output | DBMS output | Explain Plan | SQL history

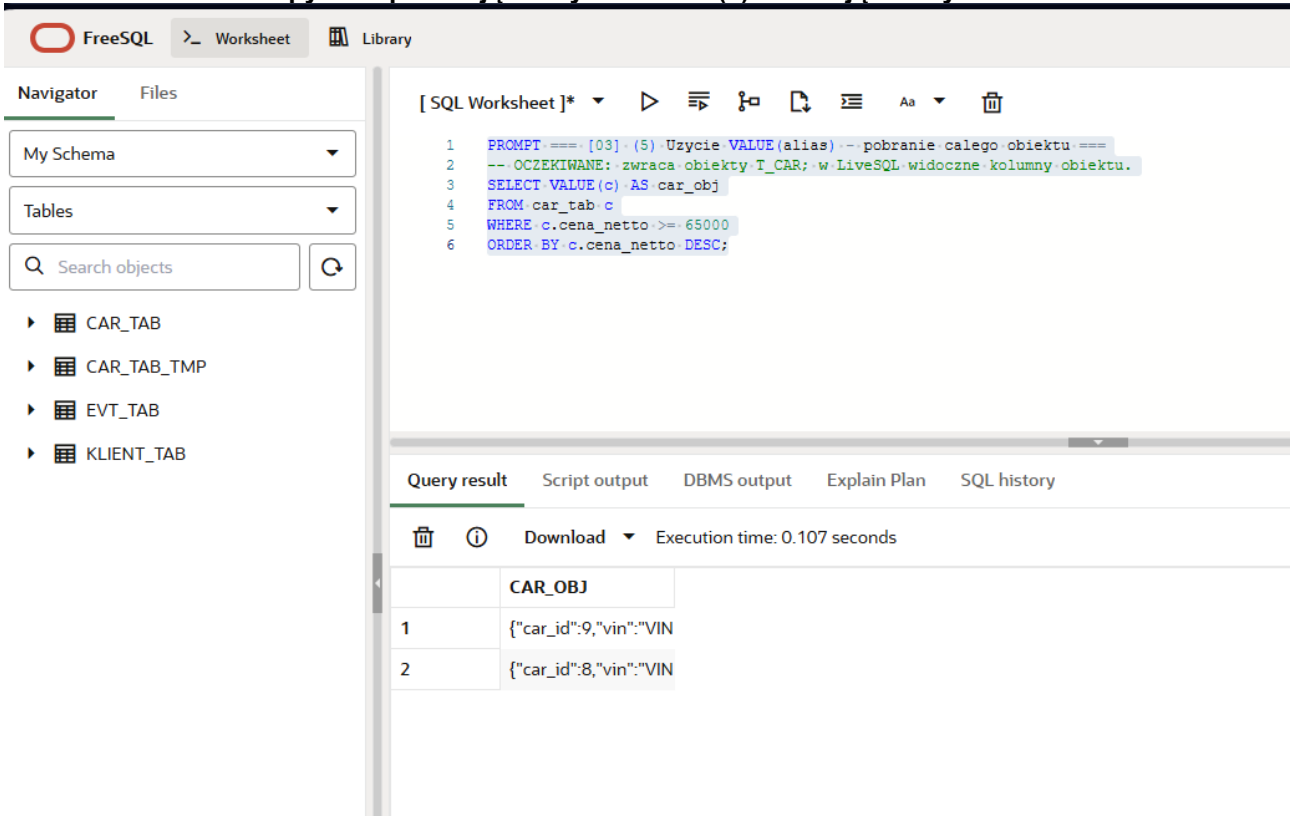
Download | Execution time: 0.012 seconds

	VIN	AUTO	LICZBA_ZDARZEN	SUMA_KOSZTOW_P
1	VIN00000000000000	Mazda 3	2	250
2	VIN00000000000000	Mazda 3	2	150
3	VIN00000000000000	Mazda 3	2	125
4	VIN00000000000000	Mazda 3	2	100
5	VIN00000000000000	Mazda 3	2	75
6	VIN00000000000000	Mazda 3	1	225
7	VIN00000000000000	Mazda 3	1	200
8	VIN00000000000000	Mazda 3	1	175
9	VIN00000000000000	Mazda 3	1	150
10	VIN00000000000000	Mazda 3	1	100

3. Filtrowanie po atrybucie auta pobranym przez DEREF (rok_prod < 2017).

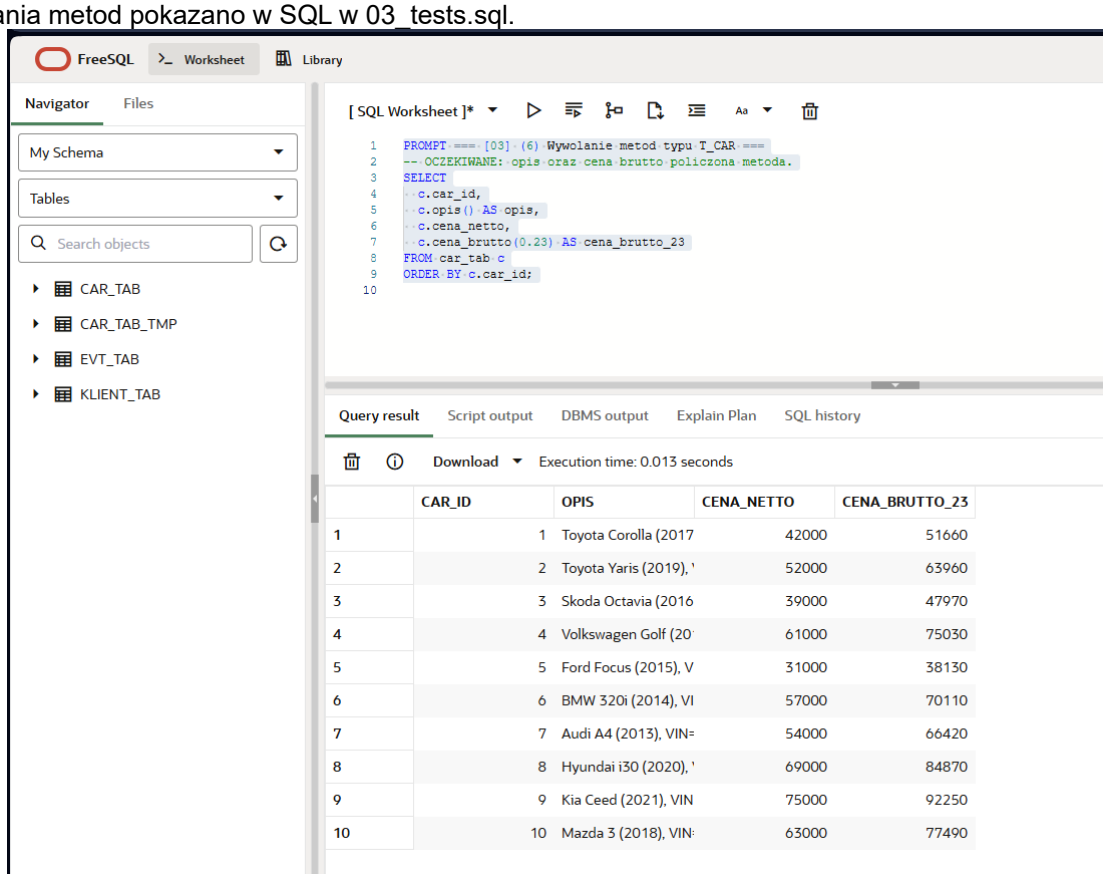


Dodatkowo: zapytanie pokazujące użycie VALUE(c) zwracające cały obiekt auta.

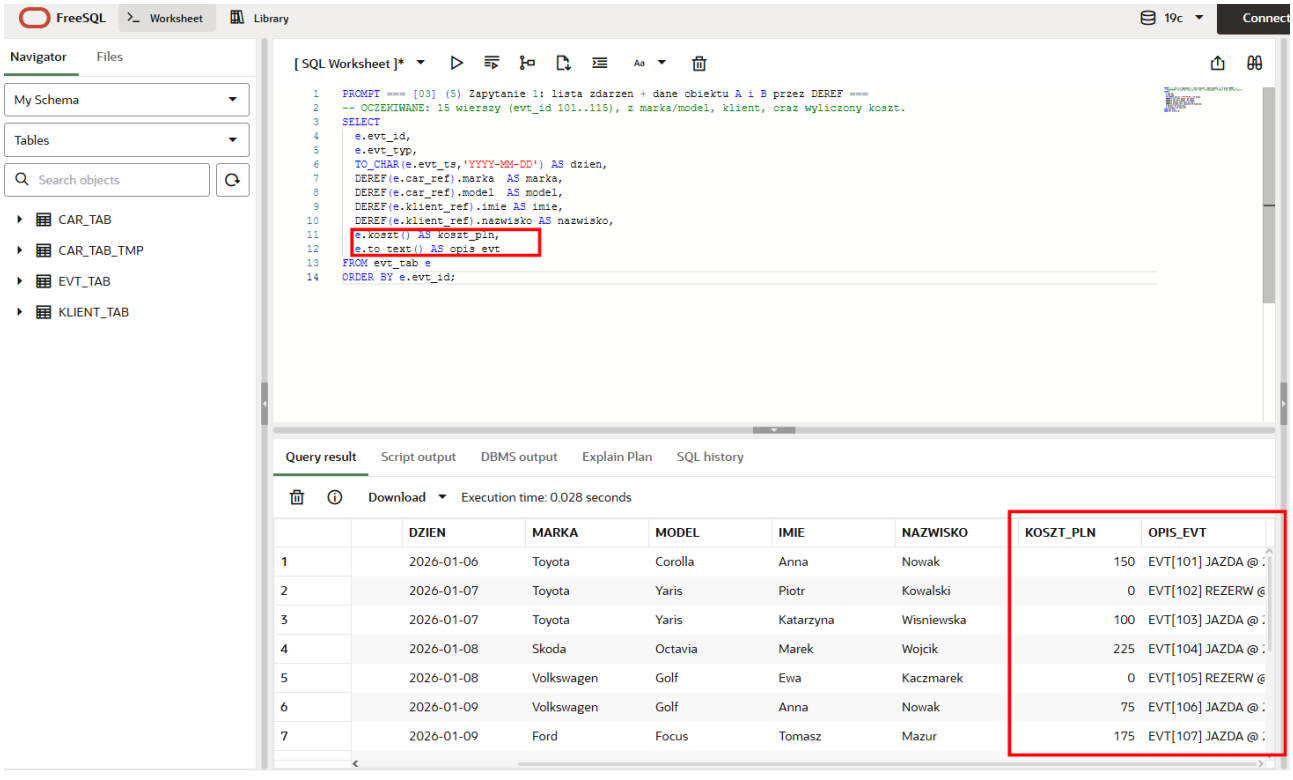


6. Metody w typach

Zaimplementowano metody w typach:
• **T_CAR.cena_brutto(p_vat)** - metoda obliczeniowa (zwrot NUMBER), wylicza cenę brutto na podstawie cena_netto.
• **T_CAR.opis()** - metoda opisowa (zwrot VARCHAR2), buduje opis auta (marka, model, rok, VIN).
Wywołania metod pokazano w SQL w 03_tests.sql.

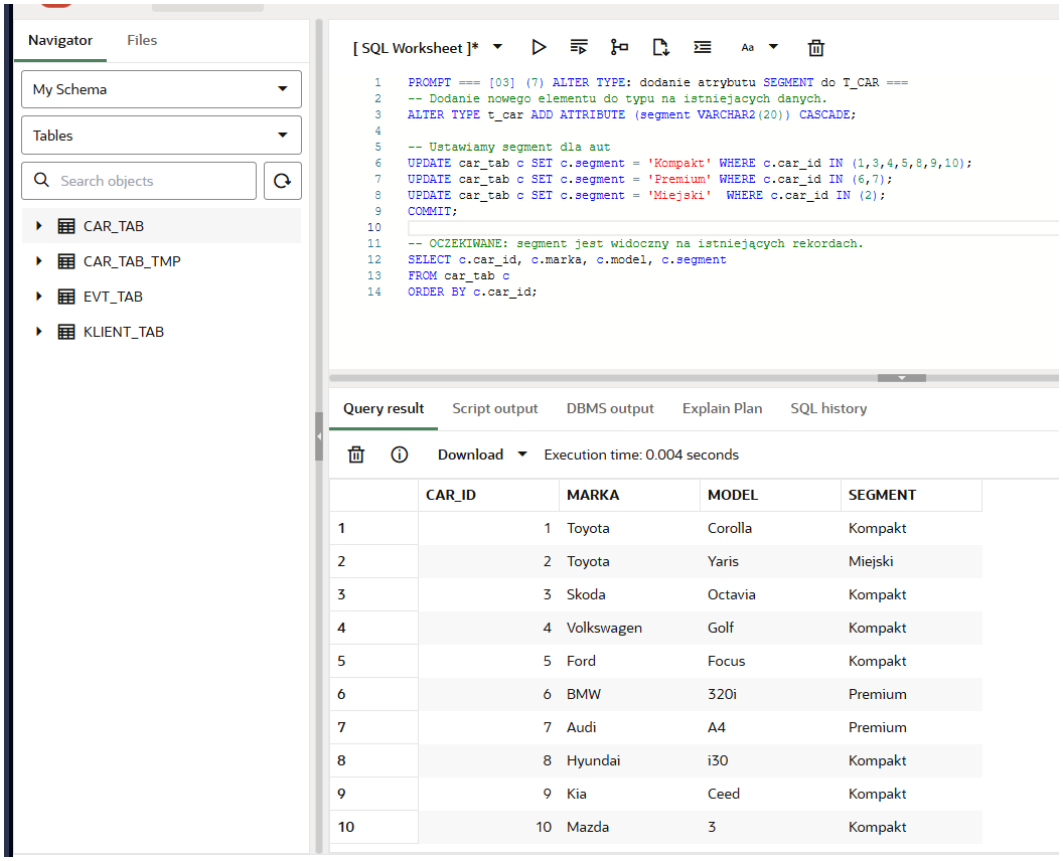


• **T_EVT.koszt()** - metoda obliczeniowa, dla typu 'JAZDA' liczy koszt (czas_min * 5), dla innych typów 0.
• **T_EVT.to_text()** - metoda opisowa, zwraca tekstowy opis zdarzenia.
Wywołania metod pokazano w SQL w 03_tests.sql.



7. ALTER TYPE

W pliku 03_tests.sql wykonano ALTER TYPE na T_CAR: dodano atrybut SEGMENT (VARCHAR2(20)) z CASCADE.
Następnie zaktualizowano istniejące rekordy w CAR_TAB, ustawiając segment dla wybranych aut, i potwierdzono działanie poprzez SELECT, ze nowy atrybut jest dostępny na istniejących danych.



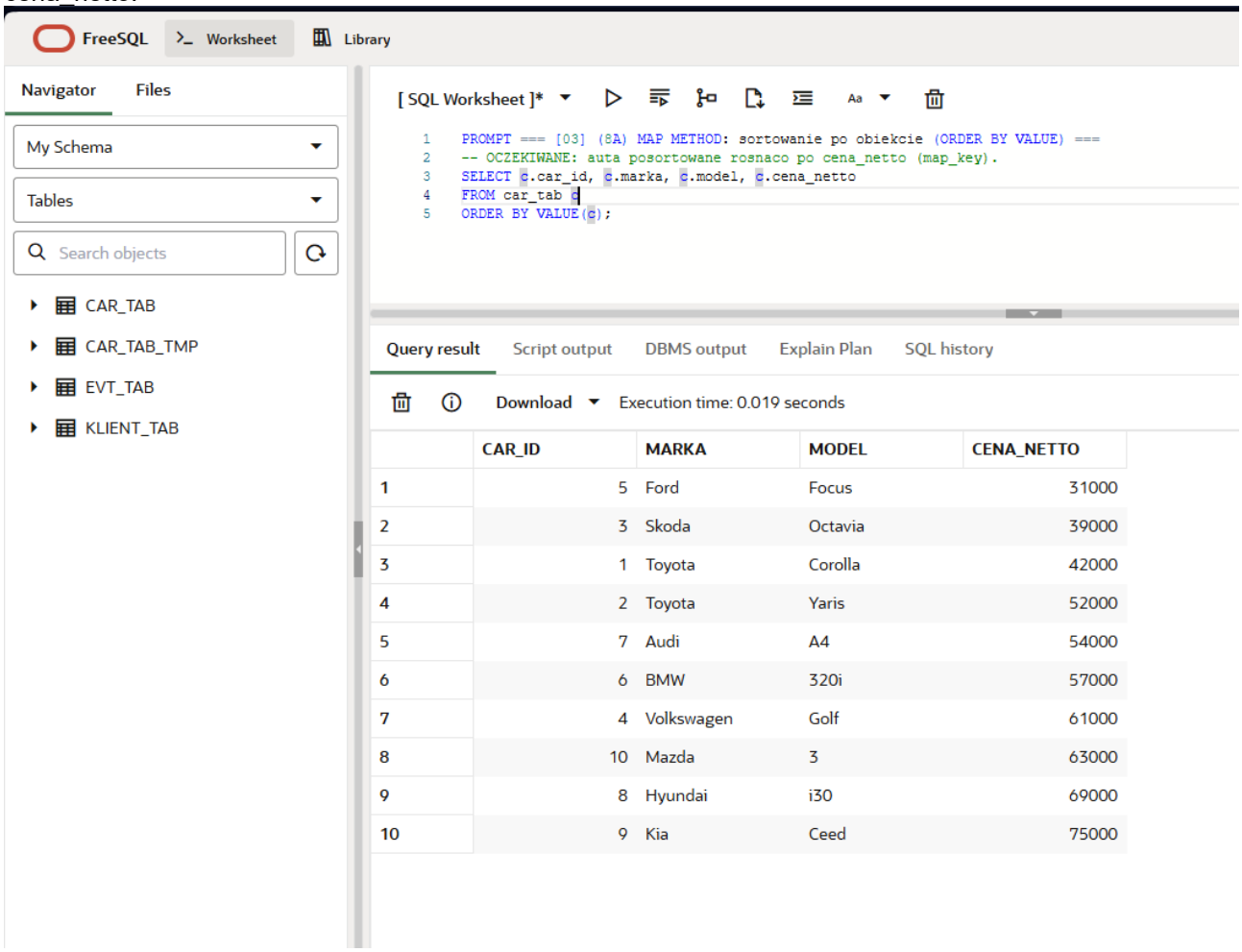
	CAR_ID	MARKA	MODEL	SEGMENT
1	1	Toyota	Corolla	Kompakt
2	2	Toyota	Yaris	Miejski
3	3	Skoda	Octavia	Kompakt
4	4	Volkswagen	Golf	Kompakt
5	5	Ford	Focus	Kompakt
6	6	BMW	320i	Premium
7	7	Audi	A4	Premium
8	8	Hyundai	i30	Kompakt
9	9	Kia	Ceed	Kompakt
10	10	Mazda	3	Kompakt

8. MAP/ORDER

Wybrano opcje A (MAP/ORDER).

W typie T_CAR dodano MAP MEMBER FUNCTION map_key RETURN NUMBER, która zwraca cena_netto. Dzięki temu Oracle może porównywać obiekty typu T_CAR podczas sortowania.

W 03_tests.sql pokazano sortowanie: ORDER BY VALUE(c), w którym posortowano auta rosnąco po cena_netto.



	CAR_ID	MARKA	MODEL	CENA_NETTO
1	5	Ford	Focus	31000
2	3	Skoda	Octavia	39000
3	1	Toyota	Corolla	42000
4	2	Toyota	Yaris	52000
5	7	Audi	A4	54000
6	6	BMW	320i	57000
7	4	Volkswagen	Golf	61000
8	10	Mazda	3	63000
9	8	Hyundai	i30	69000
10	9	Kia	Ceed	75000