

Zadanie 1 – załadowanie danych z pliku CSV

Zrzuty ekranu prezentujące proces importowania danych przy użyciu narzędzia do tego celu w Oracle SQL Developer:

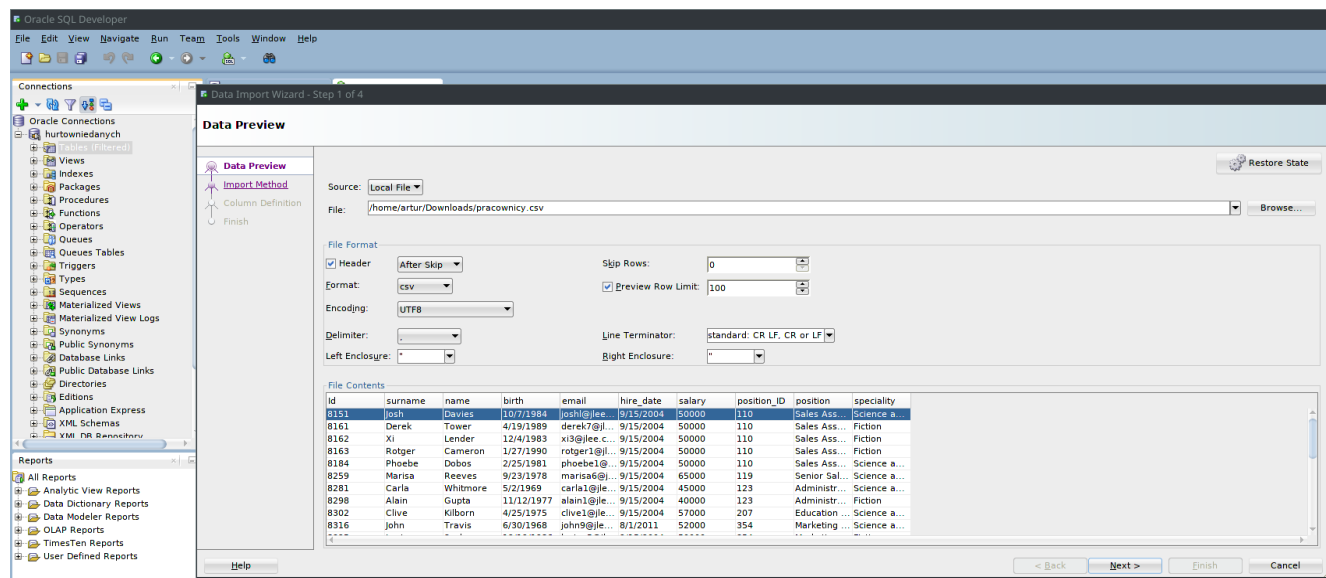


Figure 1: Wybieranie poprawnych typów danych dla zaimportowanych danych

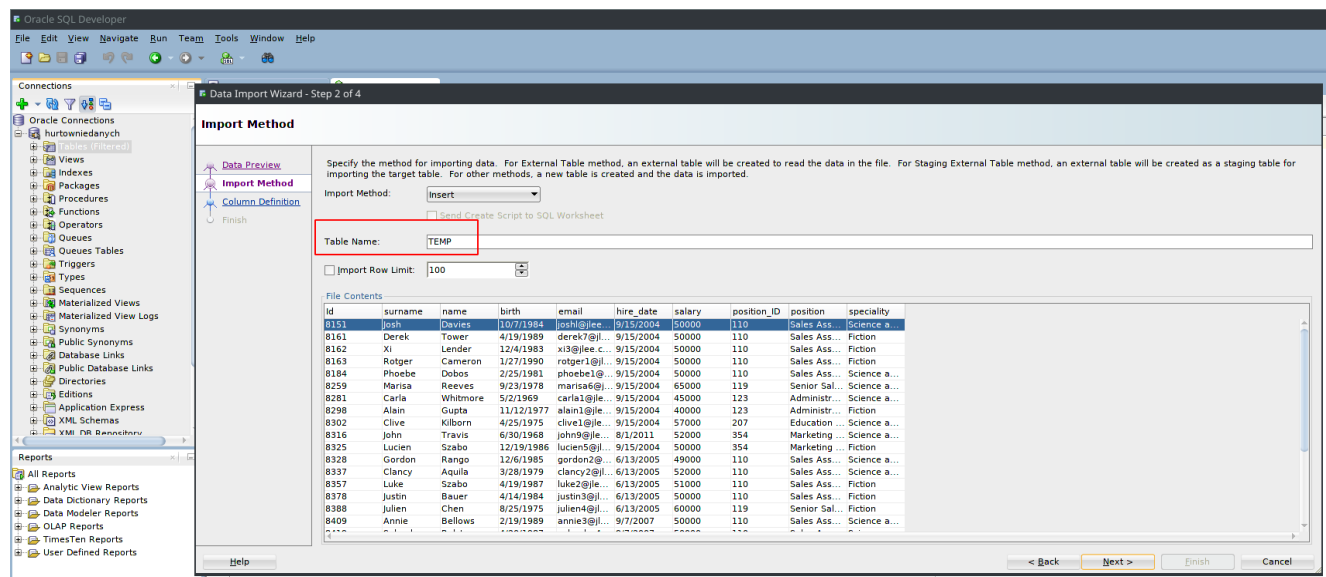


Figure 2: Tworzenie tabeli TEMP1 (nazwa TEMP była zajęta)

Connections

Oracle Connections

hurtowniedanych

Tables (Filtered)

CZAS

KOD

TEMP1

Views

hurtowniedanych.sql

hurtowniedany...

TEMP1

Columns | Data | Model | Constrai... | Grants | Statist... | Triggers | Flashback | Dependencies | Details | Partitions |

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID	NUMBER(38,0)	Yes	(null)	1 (null)	
2	SURNAME	VARCHAR2(26 BYTE)	Yes	(null)	2 (null)	
3	NAME	VARCHAR2(26 BYTE)	Yes	(null)	3 (null)	
4	BIRTH	DATE	Yes	(null)	4 (null)	

Zadanie 2 – stworzenie tabel

Kod SQL tworzący tabele Stanowisko, Specjalność oraz Pracownik:

```
CREATE TABLE stanowisko (  
    id_stanowiska    NUMBER PRIMARY KEY,  
    nazwa            VARCHAR(50)  
);  
  
CREATE TABLE specjalnosc (  
    id_specjalnosci  NUMBER PRIMARY KEY,  
    nazwa            VARCHAR(50)  
);  
  
CREATE TABLE pracownik (  
    id_pracownika    NUMBER PRIMARY KEY,  
    surname           VARCHAR(50),  
    name              VARCHAR(50),  
    birth             DATE NOT NULL,  
    hire              DATE NOT NULL,  
    email             VARCHAR(50),  
    salary            NUMBER NOT NULL,  
    id_stanowiska     REFERENCES stanowisko ( id_stanowiska ),  
    id_specjalnosci    REFERENCES specjalnosc ( id_specjalnosci )  
);
```

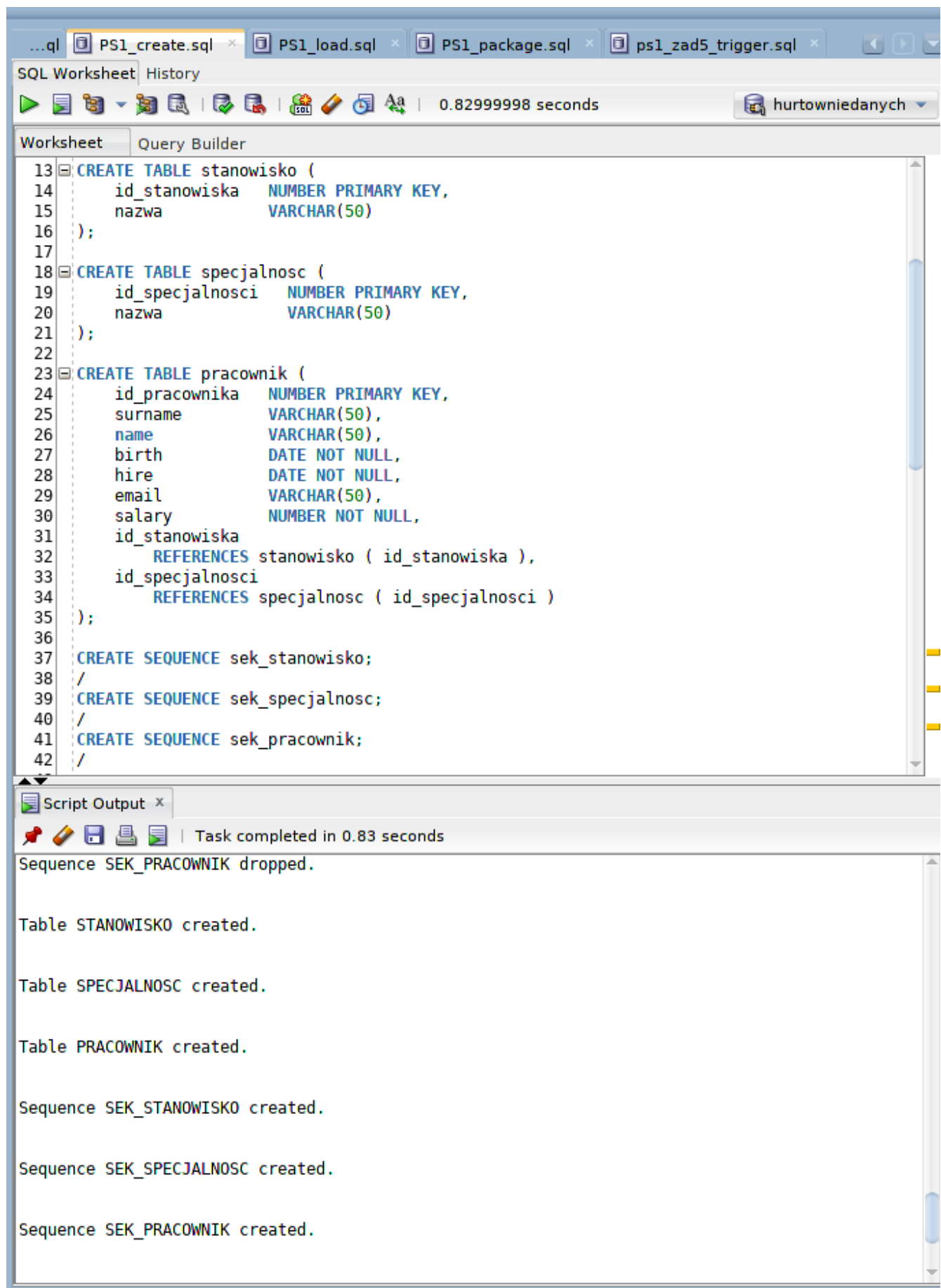


Figure 4: Wynik po uruchomieniu kodu tworzącego tabele Stanowisko, Specjalność i Pracownik

Zadanie 3 – sekwencje

Kod SQL tworzący sekwencje do auto-inkrementacji klucza głównego tabel Stanowisko, Specjalnosc I Pracownik:

```
CREATE SEQUENCE sek_stanowisko;  
/  
CREATE SEQUENCE sek_specjalnosc;  
/  
CREATE SEQUENCE sek_pracownik;  
/  
  
CREATE OR REPLACE TRIGGER t_ustaw_id_stanowisko BEFORE  
    INSERT ON stanowisko  
    FOR EACH ROW  
BEGIN  
    :new.id_stanowiska := sek_stanowisko.nextval;  
END;  
/  
CREATE OR REPLACE TRIGGER t_ustaw_id_specjalnosc BEFORE  
    INSERT ON specjalnosc  
    FOR EACH ROW  
BEGIN  
    :new.id_specjalnosci := sek_specjalnosc.nextval;  
END;  
/  
CREATE OR REPLACE TRIGGER t_ustaw_id_pracownik BEFORE  
    INSERT ON pracownik  
    FOR EACH ROW  
BEGIN  
    :new.id_pracownika := sek_pracownik.nextval;  
END;
```



...ql PS1_create.sql x PS1_load.sql x PS1_package.sql x ps1_zad5_trigger.sql x

SQL Worksheet History

| 0.82999998 seconds

hurtowniedanych ▾

Worksheet Query Builder

```
37 CREATE SEQUENCE sek_stanowisko;
38 /
39 CREATE SEQUENCE sek_specjalnosc;
40 /
41 CREATE SEQUENCE sek_pracownik;
42 /
43
44 CREATE OR REPLACE TRIGGER t_ustaw_id_stanowisko BEFORE
45     INSERT ON stanowisko
46     FOR EACH ROW
47 BEGIN
48     :new.id_stanowiska := sek_stanowisko.nextval;
49 END;
50 /
51 CREATE OR REPLACE TRIGGER t_ustaw_id_specjalnosc BEFORE
52     INSERT ON specjalnosc
53     FOR EACH ROW
54 BEGIN
55     :new.id_specjalnosci := sek_specjalnosc.nextval;
56 END;
57 /
58 CREATE OR REPLACE TRIGGER t_ustaw_id_pracownik BEFORE
59     INSERT ON pracownik
```

Zadanie 4 – pakiet (średnia pensja i ładowanie)

Kod SQL tworzący pakiet oraz ciało pakietu. Pakiet zawiera procedurę “reshape_data”, która pobiera zawartość tablicy TEMP1 po czym przekształca wiersze w taki sposób, aby uzupełnić tablice Pracownik, Specjalność oraz Stanowisko.

```
CREATE OR REPLACE PACKAGE zad4oraz5 AUTHID definer AS
    PROCEDURE reshape_data;

    FUNCTION avg_salary (
        nazwa_stanowiska VARCHAR2
    ) RETURN NUMBER;

END zad4oraz5;
/

CREATE OR REPLACE PACKAGE BODY zad4oraz5 AS

    PROCEDURE reshape_data IS
    BEGIN
        INSERT INTO specjalnosc ( nazwa )
        SELECT DISTINCT
            speciality AS name
        FROM
            temp1;

        INSERT INTO stanowisko ( nazwa )
        SELECT DISTINCT
            position AS nazwa
        FROM
            temp1;

        INSERT INTO pracownik (
            surname,
            name,
            birth,
            email,
            hire,
            salary,
            id_stanowiska,
            id_specjalnosci
        )
        SELECT
            temp1.surname,
            temp1.name,
            temp1.birth,
            temp1.email,
            temp1.hire_date,
            temp1.salary,
            stanowisko.id_stanowiska,
            specjalnosc.id_specjalnosci
        FROM
            temp1
            INNER JOIN stanowisko ON temp1.position = stanowisko.nazwa
            INNER JOIN specjalnosc ON temp1.speciality = specjalnosc.nazwa;

    END reshape_data;
```

```

FUNCTION avg_salary (
    nazwa_stanowiska VARCHAR2
) RETURN NUMBER IS
    srednia_pensja NUMBER;
BEGIN
    SELECT
        avgsal
    INTO srednia_pensja
    FROM
        (
            SELECT
                AVG(pracownik.salary) AS avgsal,
                pracownik.id_stanowiska AS idst
            FROM
                pracownik
            GROUP BY
                pracownik.id_stanowiska
        )
    INNER JOIN stanowisko ON stanowisko.id_stanowiska = idst
    WHERE
        stanowisko.nazwa LIKE nazwa_stanowiska;

    return(srednia_pensja);
END;

END zad4oraz5;
/

```


SQL Worksheet History

0.347 seconds

hurtowniedanych

Worksheet Query Builder

```
) RETURN NUMBER IS
  srednia_pensja NUMBER;
BEGIN
  SELECT
    avgsal
  INTO srednia_pensja
  FROM
    (
      SELECT
        AVG(pracownik.salary) AS avgsal,
        pracownik.id_stanowiska AS idst
      FROM
        pracownik
      GROUP BY
        pracownik.id_stanowiska
    )
    INNER JOIN stanowisko ON stanowisko.id_stanowiska = idst
  WHERE
    stanowisko.nazwa LIKE nazwa_stanowiska;

  return(srednia_pensja);
END;
```

END zad4oraz5;

Script Output x Query Result x Query Result 1 x

Task completed in 0.347 seconds

```

78 FROM
79     specjalnosc;
80
81 EXECUTE zad4oraz5.reshape_data();
82
83 zad4oraz5.avg_salary('Education Analyst');
84
85 SELECT * FROM temp1;
86 SELECT * from pracownik;
87 SELECT * FROM specjalnosc;
88 SELECT * from stanowisko;
89
90 SELECT * FROM V$SESSION WHERE STATUS = 'ACTIVE';

```

Script Output x

Task completed in 0.172 seconds

Sequence SEK_PRACOWNIK created.

Trigger T_USTAW_ID_STANOWISKO compiled

Trigger T_USTAW_ID_SPECJALNOSC compiled

Trigger T_USTAW_ID_PRACOWNIK compiled

PL/SQL procedure successfully completed.

Figure 7: Wynik wywołania procedury uzupełniającej tabele Stanowisko, Pracownik i Specjalność na podstawie tabeli TEMP1 (zawierającej dane z pliku pracownicy.csv). Procedura nazywa się "reshape_data"

```

30
31 SELECT
32     zad4oraz5.avg_salary('Education Analyst')
33 FROM
34     dual;
35
36 EXECUTE zad4oraz5.reshape_data();

```

Script Output x

Query Result x

All Rows Fetched: 1 in 0.019 seconds

ZAD4ORAZ5.AVG_SALARY('EDUCATIONANALYST')	
1	57000

Figure 8: Wynik wywołania funkcji "avg_salary", która zwraca średnią pensję na danym stanowisku. Nazwa stanowiska jest podawana do funkcji poprzez parametr. W tym wypadku badanym stanowiskiem jest 'Education Analyst', którego średnia pensja wynosi 57000

Zadanie 5 – wyzwalacze

Kod SQL tworzący wyzwalacze, które zabraniają dodania (lub zmianę - UPDATE) użytkownika z pensją większą niż 30% od średniej (poniżej). Wyzwalacze używają wewnątrz funkcji “avg_salary” z poprzedniego zadania.

```
CREATE OR REPLACE TRIGGER ogranicz_pensje_insert BEFORE
    INSERT ON pracownik
    FOR EACH ROW
DECLARE
    avg_salary_result NUMBER;
BEGIN
    SELECT
        zad4oraz5.avg_salary(st.nazwa)
    INTO avg_salary_result
    FROM
        stanowisko st
    WHERE
        st.id_stanowiska = :new.id_stanowiska;

    IF ( :new.salary > 1.3 * avg_salary_result ) THEN
        raise_application_error(-20001, 'Pensja nie może być większa o 30% od
średniej pensji na danym stanowisku');
    END IF;

END;
/
CREATE OR REPLACE TRIGGER ogranicz_pensje_update BEFORE
    UPDATE ON pracownik
    FOR EACH ROW
DECLARE
    avg_salary_result NUMBER;
BEGIN
    SELECT
        zad4oraz5.avg_salary(st.nazwa)
    INTO avg_salary_result
    FROM
        stanowisko st
    WHERE
        st.id_stanowiska = :new.id_stanowiska;

    IF ( :new.salary > 1.3 * avg_salary_result ) THEN
        raise_application_error(-20001, 'Pensja nie może być większa o 30% od
średniej pensji na danym stanowisku');
    END IF;

END;

INSERT INTO tagisow_artur.pracownik (
    surname,
    name,
    birth,
    hire,
    email,
    salary,
    id_stanowiska,
    id_specjalnosci
) VALUES (
```

```

        'Julien',
        'Chen',
        TO_DATE('25-AUG-75', 'DD-MON-RR'),
        TO_DATE('13-JUN-05', 'DD-MON-RR'),
        'julien4@jlee.com',
        999999,
        21,
        21
    );
/

```

UPDATE pracownik

SET

```

    surname = 'Julien',
    name = 'Chen',
    birth = TO_DATE('25-AUG-75', 'DD-MON-RR'),
    hire = TO_DATE('13-JUN-05', 'DD-MON-RR'),
    email = 'julien4@jlee.com',
    salary = 999999,
    id_stanowiska = 21,
    id_specjalnosci = 21

```

WHERE

```

    pracownik.id_pracownika = 41;
--      AND pracownik.birth LIKE TO_DATE('25-AUG-75', 'DD-MON-RR');
/

```

SQL Worksheet | History | 0.33399999 seconds | hurtowniedanych

Worksheet | Query Builder

```
raise_application_error(-20001, 'Pensja nie może być większa o 30% od średniej')
END IF;

END;
/
CREATE OR REPLACE TRIGGER ogranicz_pensje_update BEFORE
UPDATE ON pracownik
FOR EACH ROW
DECLARE
avg_salary_result NUMBER;
BEGIN
SELECT
    zad4oraz5.avg_salary(st.nazwa)
INTO avg_salary_result
FROM
    stanowisko st
WHERE
    st.id_stanowiska = :new.id_stanowiska;

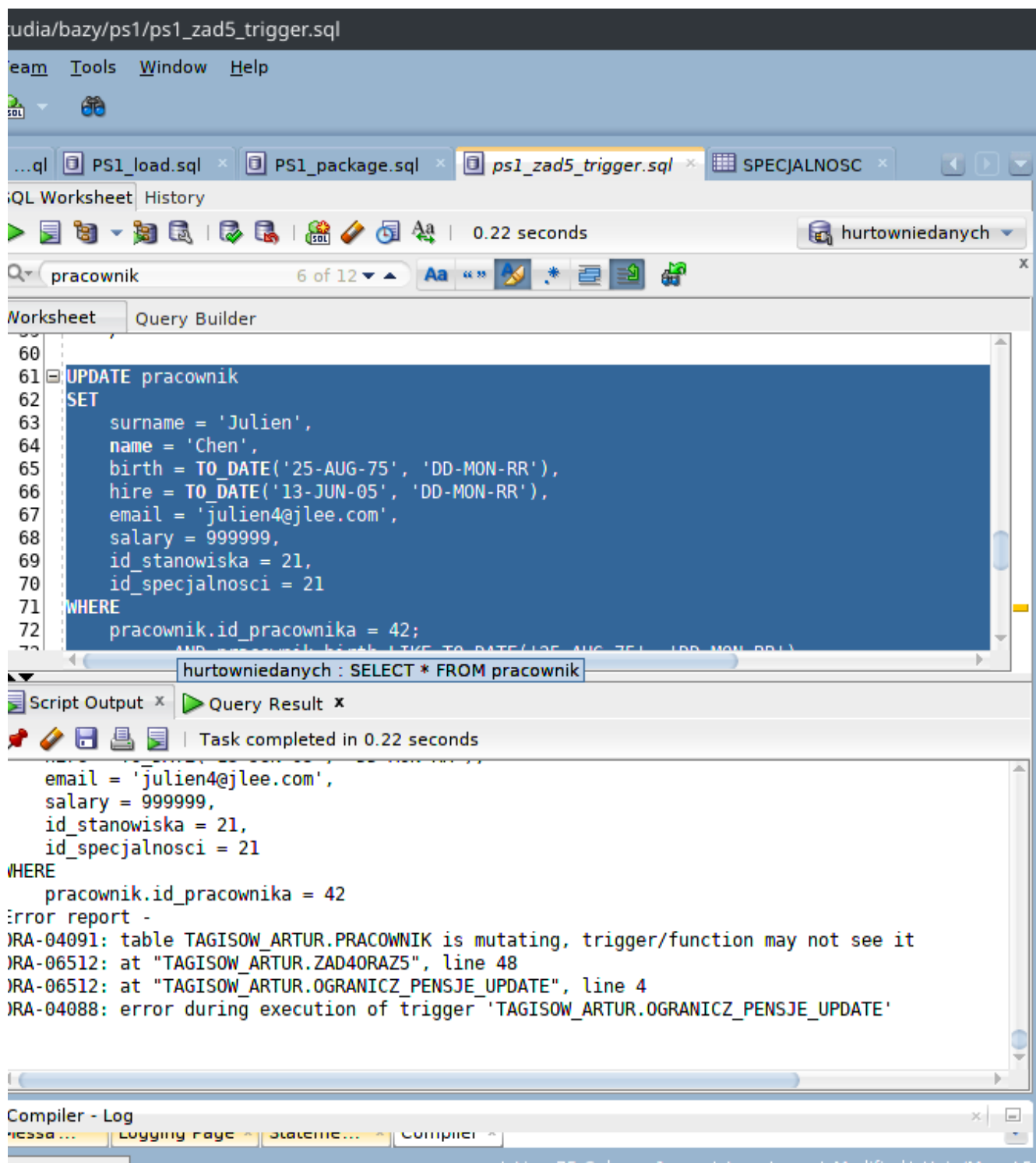
IF ( :new.salary > 1.3 * avg_salary_result ) THEN
    raise_application_error(-20001, 'Pensja nie może być większa o 30% od średniej')
END IF;
END;

INSERT INTO pracownik (
```

Script Output x | Task completed in 0.334 seconds

Trigger OGRANICZ_PENSJE_INSERT compiled

Trigger OGRANICZ_PENSJE_UPDATE compiled



Zrzuty ekranu tabel pracownik, specjalizacja, stanowisko (id mogą nieco się różnić w niektórych screenach – czasami id specjalności mogły mieć id 21 I 22.

```

86 SELECT * FROM pracownik;
87 SELECT * FROM specjalnosc;
88 SELECT * FROM stanowisko;
89
90 SELECT * FROM V$SESSION WHERE STATUS = 'ACTIVE';

```

Script Output x Query Result x

SQL | All Rows Fetched: 27 in 0.027 seconds

ID_PRACOWNIKA	SURNAME	NAME	BIRTH	HIRE	EMAIL	SALARY	ID_STANOWISKA	ID_SPECJALNOSCI
1	Julien	Chen	25-AUG-75	13-JUN-05	julien4@jlee.com	60000	1	1
2	Marisa	Reeves	23-SEP-78	15-SEP-04	marisa6@jlee.com	65000	1	2
3	Gloria	Goodman	21-JUN-65	01-JUN-11	gloria5@jlee.com	60000	2	2
4	Adam	Wells	05-JUL-91	01-AUG-11	adamw@jlee.com	46000	3	2

```

87 SELECT * FROM specjalnosc;
88 SELECT * FROM stanowisko;
89
90 SELECT * FROM V$SESSION WHERE STATUS = 'ACTIVE';

```

Script Output x Query Result x

SQL | All Rows Fetched: 2 in 0.027 seconds

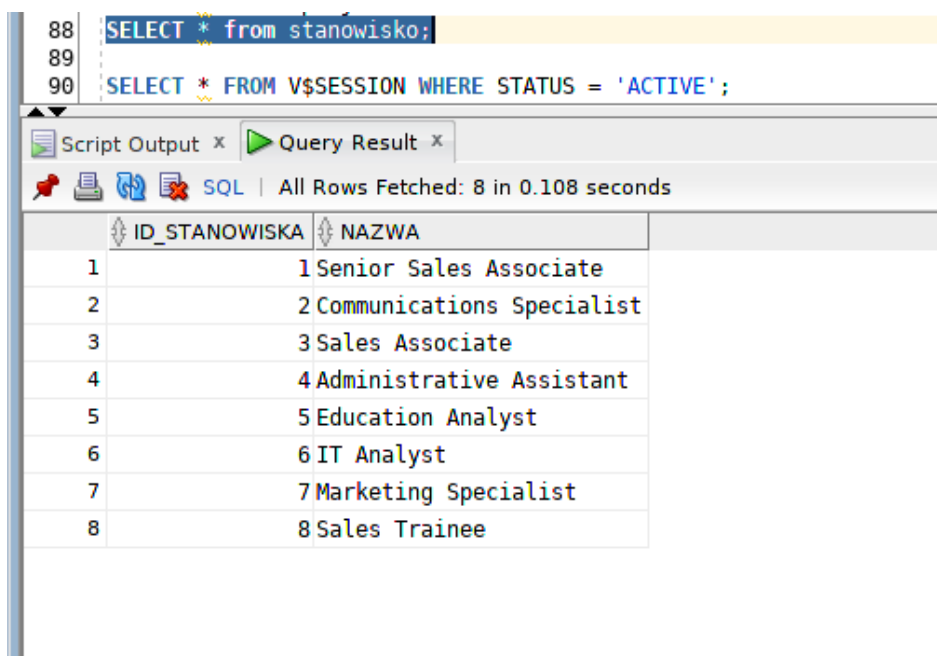
ID_SPECJALNOSCI	NAZWA
1	1 Fiction
2	2 Science and Engineering

Figure 10: Zrzut ekranu tabeli Specjalnosc

Wnioski

Realizacja wszystkich zadań powiodła się z wyjątkiem:

- Zadania 5 – wyzwalacz “before update” - z nieznanego powodu wywołanie polecenia UPDATE na tabeli po założeniu tego wyzwalacza zawsze otrzymano błąd o tym, że tabela PRACOWNIK jest aktualnie modyfikowana, niezależnie od momentu w którym momencie zostało ono wykonane (np. Po godzinie od ostatniej operacji CREATE, INSERT itp.)
- Zadania 3 – skryptu ładującego przy użyciu MERGE



The screenshot shows the Oracle SQL Developer interface. At the top, a SQL query is entered in the editor: `SELECT * from stanowisko;` on line 88 and `SELECT * FROM V$SESSION WHERE STATUS = 'ACTIVE';` on line 90. Below the editor, the 'Query Result' tab is active, displaying a table with 8 rows. The table has two columns: 'ID_STANOWISKA' and 'NAZWA'. The data rows are numbered 1 through 8, corresponding to the 'ID_STANOWISKA' values. The job titles are: 1 Senior Sales Associate, 2 Communications Specialist, 3 Sales Associate, 4 Administrative Assistant, 5 Education Analyst, 6 IT Analyst, 7 Marketing Specialist, and 8 Sales Trainee. The status bar indicates 'All Rows Fetched: 8 in 0.108 seconds'.

ID_STANOWISKA	NAZWA
1	1 Senior Sales Associate
2	2 Communications Specialist
3	3 Sales Associate
4	4 Administrative Assistant
5	5 Education Analyst
6	6 IT Analyst
7	7 Marketing Specialist
8	8 Sales Trainee

Figure 11: Zrzut ekranu tabeli Stanowisko

Oracle SQL Developer jest dość mało stabilnym programem na systemie operacyjnym Linux. W niektórych przypadkach może to spowodować stworzenie niezamkniętej sesji blokującej dostęp do pewnej tabeli.

Praca na uczelnianym serwerze Oracle jest o tyle trudniejsza, że nie istnieje możliwość ręcznego zamknięcia takiej sesji. W takich przypadkach potrzebna jest pomoc administratora.

Warto dodać, że z nieznanego powodu, najmniejszym ID pracownika było np. 21, kiedy oczekiwaną pierwszą wartością jest oczywiście 1. Problem pojawiał się oraz znikał podczas kolejnych importów danych z tabeli TEMP1. Nie jest to szkodliwe zjawisko. Jednak jest na pewno niespodziewane.