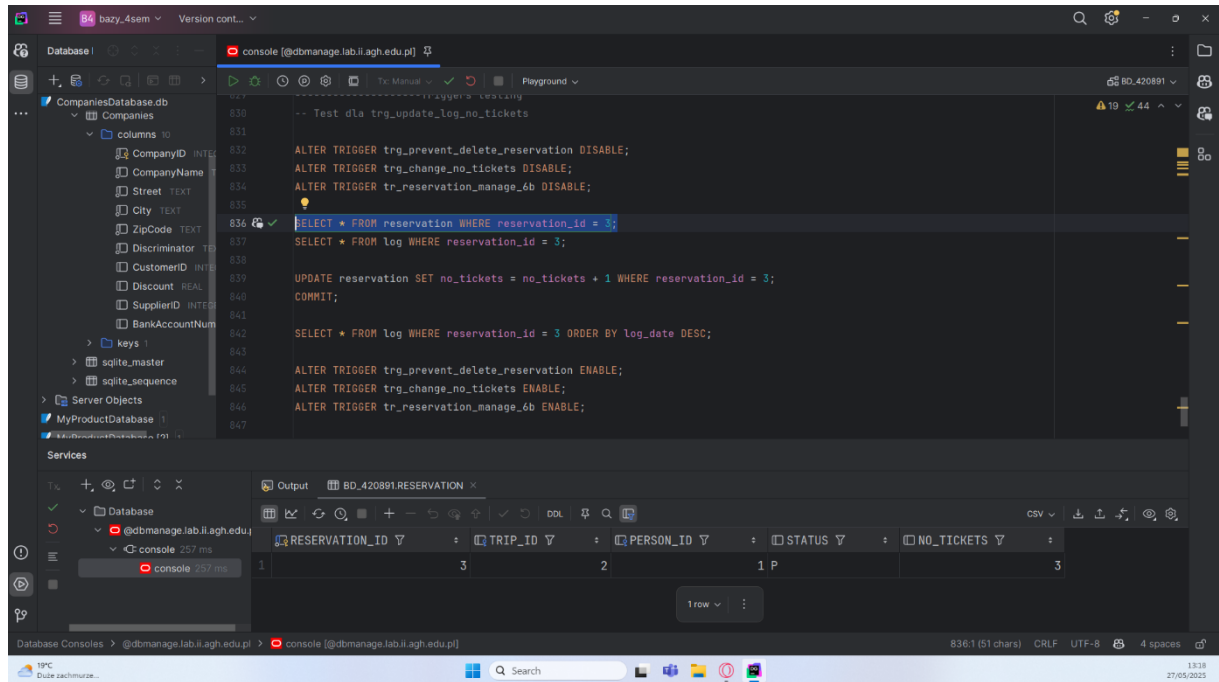


Test triggera trg\_update\_log\_no\_tickets:



The screenshot shows a SQL Server Enterprise Manager interface. The left pane displays the database structure for 'CompaniesDatabase.db'. The center pane shows a SQL script with the following commands:

```
-- Test dla trg_update_log_no_tickets
ALTER TRIGGER trg_prevent_delete_reservation DISABLE;
ALTER TRIGGER trg_change_no_tickets DISABLE;
ALTER TRIGGER tr_reservation_manage_6b DISABLE;

SELECT * FROM reservation WHERE reservation_id = 3;
SELECT * FROM log WHERE reservation_id = 3;

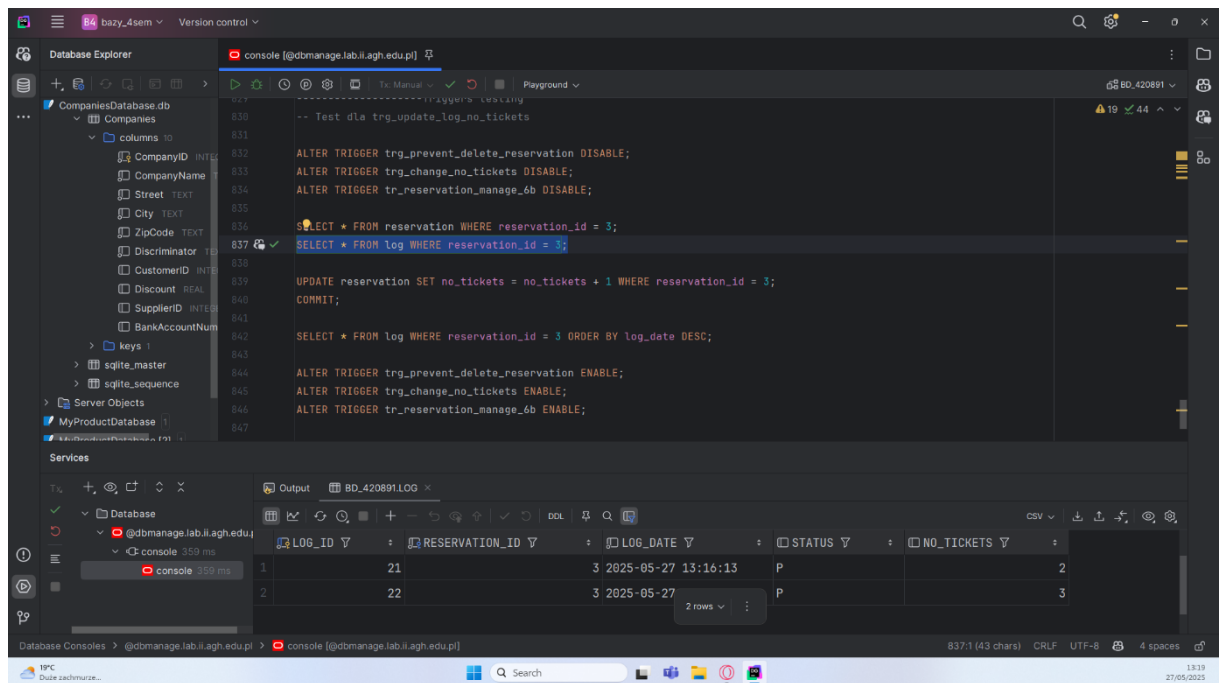
UPDATE reservation SET no_tickets = no_tickets + 1 WHERE reservation_id = 3;
COMMIT;

SELECT * FROM log WHERE reservation_id = 3 ORDER BY log_date DESC;

ALTER TRIGGER trg_prevent_delete_reservation ENABLE;
ALTER TRIGGER trg_change_no_tickets ENABLE;
ALTER TRIGGER tr_reservation_manage_6b ENABLE;
```

The right pane shows the output of the script, displaying the reservation details for reservation\_id 3:

RESERVATION_ID	TRIP_ID	PERSON_ID	STATUS	NO_TICKETS
3	2	1	P	3



The screenshot shows the same SQL Server Enterprise Manager interface. The SQL script is the same as in the previous screenshot. The output pane now shows the log entries for the reservation update:

LOG_ID	RESERVATION_ID	LOG_DATE	STATUS	NO_TICKETS
21	3	2025-05-27 13:16:13	P	2
22	3	2025-05-27	P	3

The screenshot shows the SQL Developer interface with a script being executed in the console. The script includes SQL commands to disable triggers, select data from the reservation\_log table, update the no\_tickets column, and then re-enable the triggers. The output window shows the results of the SELECT statement, displaying reservation details.

```

827 -----Trigger's Testing
828 -- Test dla trg_update_log_no_tickets
829
830 ALTER TRIGGER trg_prevent_delete_reservation DISABLE;
831 ALTER TRIGGER trg_change_no_tickets DISABLE;
832 ALTER TRIGGER tr_reservation_manage_6b DISABLE;
833
834 SELECT * FROM reservation WHERE reservation_id = 3;
835 SELECT * FROM log WHERE reservation_id = 3;
836
837 UPDATE reservation SET no_tickets = no_tickets + 1 WHERE reservation_id = 3;
838 COMMIT;
839
840 SELECT * FROM log WHERE reservation_id = 3 ORDER BY log_date DESC;
841
842 ALTER TRIGGER trg_prevent_delete_reservation ENABLE;
843 ALTER TRIGGER trg_change_no_tickets ENABLE;
844 ALTER TRIGGER tr_reservation_manage_6b ENABLE;
845
846
847

```

LOG_ID	RESERVATION_ID	LOG_DATE	STATUS	NO_TICKETS
23	3	2025-05-27 13:19:11	P	4
22	3	2025-05-27	P	3
21	3	2025-05-27	P	2

Test triggera trg\_prevent\_delete\_reservation:

The screenshot shows a PL/SQL block being executed in the console. The block attempts to delete a reservation with reservation\_id = 3, which is prevented by the trg\_prevent\_delete\_reservation trigger. The output window shows the execution details and the error message raised by the trigger.

```

852
853 --test dla trg_prevent_delete_reservation
854
855 ALTER TRIGGER trg_update_log_no_tickets DISABLE;
856 ALTER TRIGGER trg_change_no_tickets DISABLE;
857 ALTER TRIGGER tr_reservation_manage_6b DISABLE;
858
859 BEGIN
860     DELETE FROM reservation WHERE reservation_id = 3;
861     COMMIT;
862 EXCEPTION
863     WHEN OTHERS THEN
864         DBMS_OUTPUT.PUT_LINE('Expected error: ' || SQLERRM);
865 END;
866
867 ALTER TRIGGER trg_update_log_no_tickets ENABLE;
868 ALTER TRIGGER trg_change_no_tickets ENABLE;
869 ALTER TRIGGER tr_reservation_manage_6b ENABLE;
870

```

[2025-05-27 13:21:20] completed in 14 ms  
Expected error: ORA-20001: You cannot delete reservation. You can only cancel it.  
ORA-06512: at "BD\_420891.TRG\_PREVENT\_DELETE\_RESERVATION", line 2  
ORA-04688: error during execution of trigger 'BD\_420891.TRG\_PREVENT\_DELETE\_RESERVATION'

## Test dla trg\_check\_reservation\_availability

The screenshot shows the SQL Developer interface with a PL/SQL script named `check_reser` in the console. The script includes a `SELECT` statement to check for reservations with fewer than 5 available places, followed by a `BEGIN` block that calls `p_add_reservation_5` with specific parameters. An `EXCEPTION` block handles errors by displaying the `SQLERRM`.

```
880
881 SELECT * FROM vw_trip WHERE no_available_places < 5;
882 SELECT RESERVATION_ID, TRIP_NAME, TRIP_ID, FIRSTNAME, LASTNAME, STATUS, NO_TICKETS FROM VW_RESERVATION where person_id = 1;
883
884 BEGIN
885     p_add_reservation_5(
886         p_trip_id => 2,
887         p_person_id => 1,
888         p_no_tickets => 100 + 10
889     );
890 EXCEPTION
891     WHEN OTHERS THEN
892         DBMS_OUTPUT.PUT_LINE('Expected error: ' || SQLERRM);
893 END;
894 /
```

The output window displays the results of the `VW_TRIP` view query:

TRIP_ID	COUNTRY	TRIP_DATE	TRIP_NAME	MAX_NO_PLACES	NO_AVAILABLE_PLACES
1	Francja	2023-09-12	Wycieczka do Paryza	9	1
4	Polska	2025-05-01	Hot	4	4
2	Polska	2025-05-03	Piekny Krakow	6	0

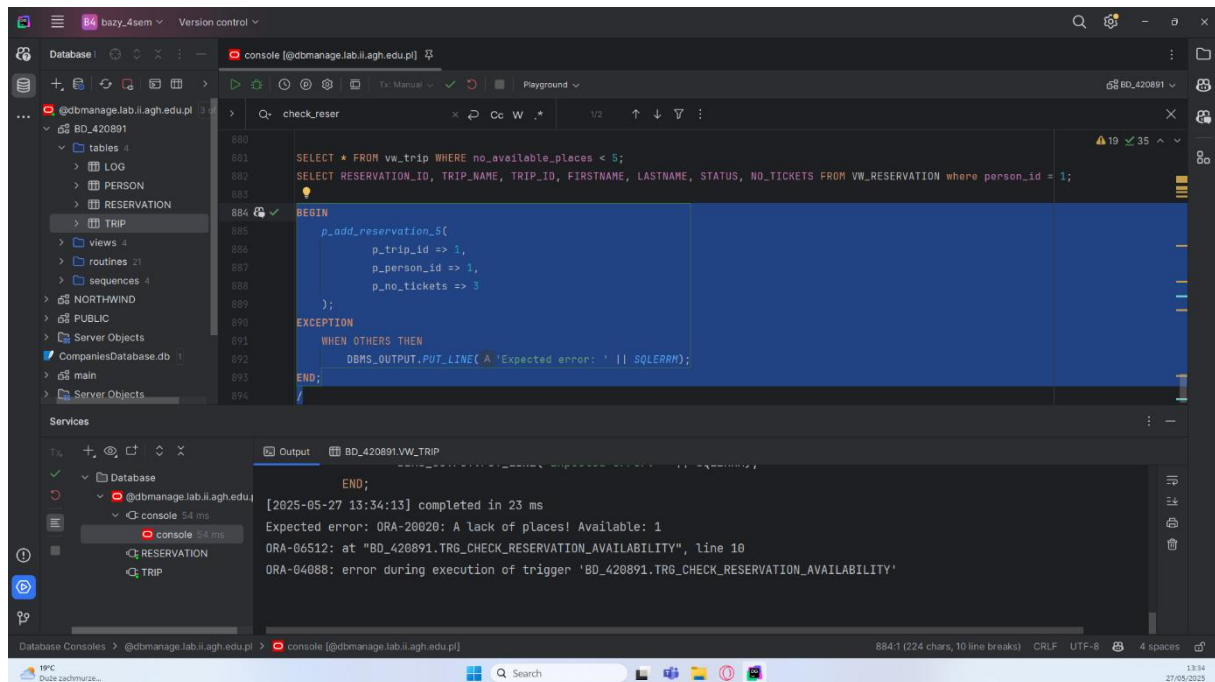
This screenshot shows the same SQL Developer interface, but the output window displays the results of the `VW_RESERVATION` view query. The script is identical to the one in the first screenshot.

```
880
881 SELECT * FROM vw_trip WHERE no_available_places < 5;
882 SELECT RESERVATION_ID, TRIP_NAME, TRIP_ID, FIRSTNAME, LASTNAME, STATUS, NO_TICKETS FROM VW_RESERVATION where person_id = 1;
883
884 BEGIN
885     p_add_reservation_5(
886         p_trip_id => 2,
887         p_person_id => 1,
888         p_no_tickets => 100 + 10
889     );
890 EXCEPTION
891     WHEN OTHERS THEN
892         DBMS_OUTPUT.PUT_LINE('Expected error: ' || SQLERRM);
893 END;
894 /
```

The output window displays the results of the `VW_RESERVATION` view query:

RESERVATION_ID	TRIP_NAME	TRIP_ID	FIRSTNAME	LASTNAME	STATUS	NO_TICKETS
1	Wycieczka do Paryza	1	Jan	Nowak	P	1
2	Piekny Krakow	2	Jan	Nowak	P	4

Wybieram trip\_id = 1:



The screenshot shows the SQL Developer interface with a PL/SQL procedure named `p_add_reservation_5` being executed. The procedure is called with `p_trip_id => 1`, `p_person_id => 1`, and `p_no_tickets => 3`. The output window shows the execution results, including a message indicating that the reservation was added successfully and that there are 1 available places left.

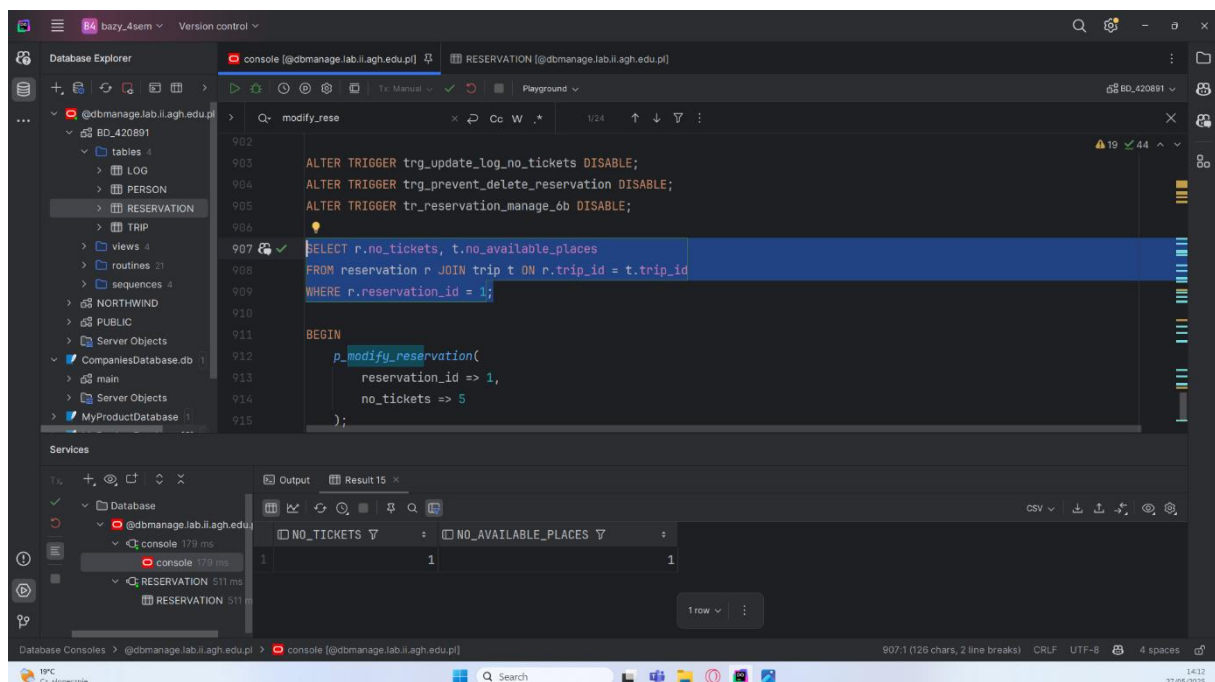
```
SELECT * FROM vw_trip WHERE no_available_places < 5;
SELECT RESERVATION_ID, TRIP_NAME, TRIP_ID, FIRSTNAME, LASTNAME, STATUS, NO_TICKETS FROM VW_RESERVATION where person_id = 1;

BEGIN
  p_add_reservation_5(
    p_trip_id => 1,
    p_person_id => 1,
    p_no_tickets => 3
  );
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Expected error: ' || SQLERRM);
END;
```

Output:

```
[2025-05-27 13:34:13] completed in 23 ms
Expected error: ORA-20020: A lack of places! Available: 1
ORA-06512: at "BD_420891.TRG_CHECK_RESERVATION_AVAILABILITY", line 10
ORA-04088: error during execution of trigger 'BD_420891.TRG_CHECK_RESERVATION_AVAILABILITY'
```

test dla trg\_change\_no\_tickets:



The screenshot shows the SQL Developer interface with a SQL query being executed. The query selects the reservation ID, trip name, trip ID, first name, last name, status, and number of tickets from the `VW_RESERVATION` view, filtered by `person_id = 1`. The output window shows the results of the query, displaying a single row with the reservation ID 1, trip name 'NORTHWIND', trip ID 1, first name 'ALEXANDER', last name 'BERNSTEIN', status 'OK', and number of tickets 1.

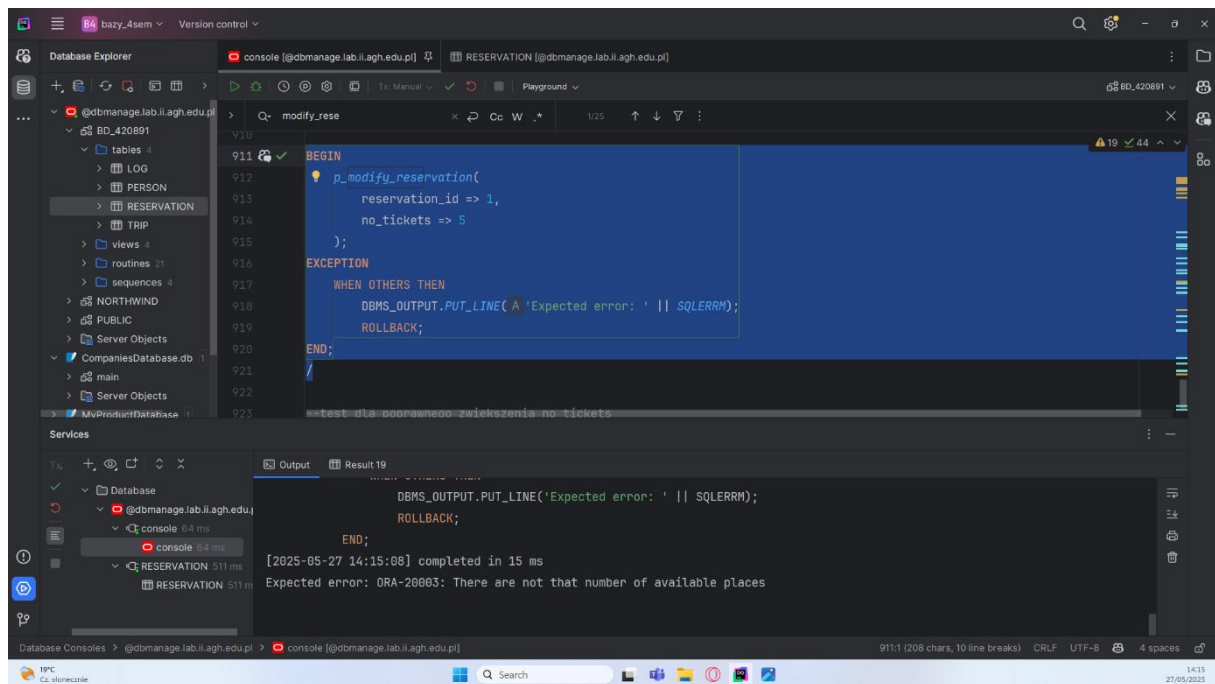
```
ALTER TRIGGER trg_update_log_no_tickets DISABLE;
ALTER TRIGGER trg_prevent_delete_reservation DISABLE;
ALTER TRIGGER trg_reservation_manage_6b DISABLE;

SELECT r.no_tickets, t.no_available_places
FROM reservation r JOIN trip t ON r.trip_id = t.trip_id
WHERE r.reservation_id = 1;

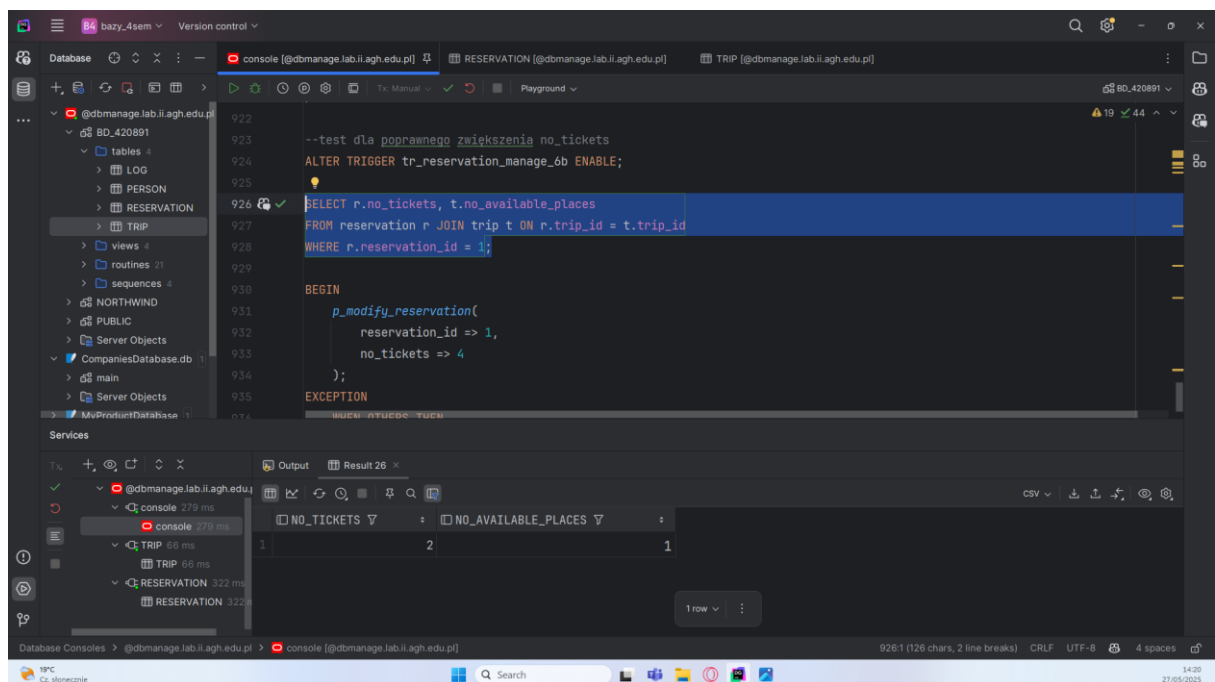
BEGIN
  p_modify_reservation(
    reservation_id => 1,
    no_tickets => 5
  );
END;
```

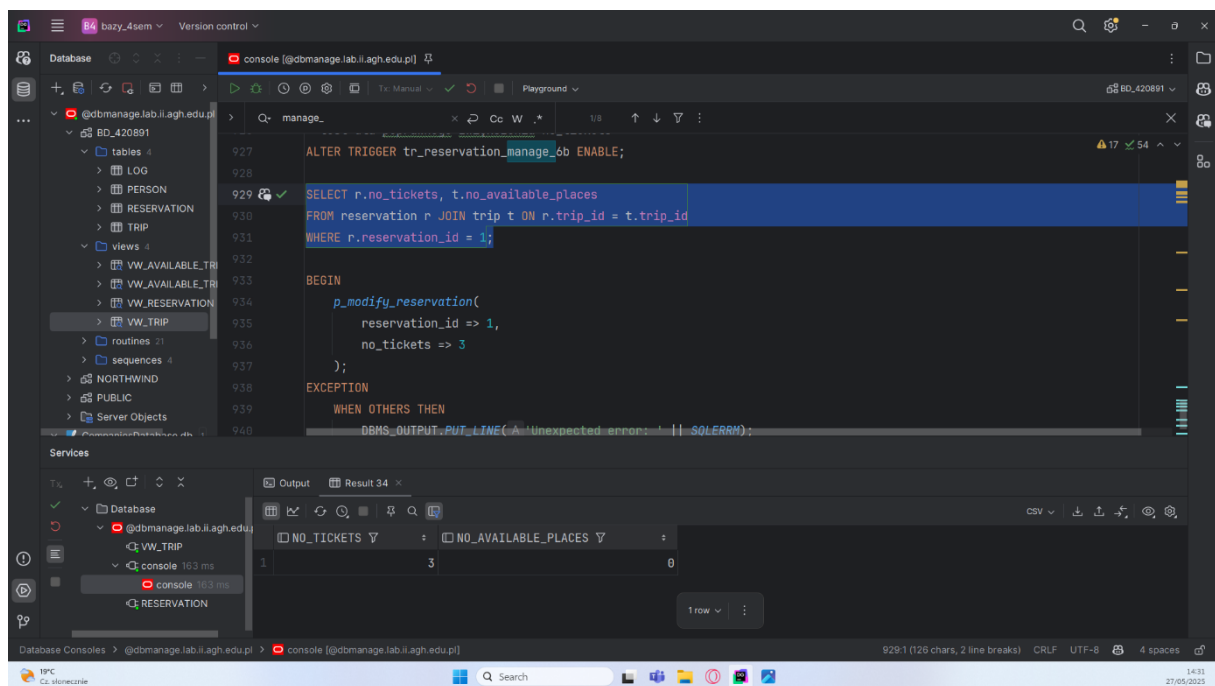
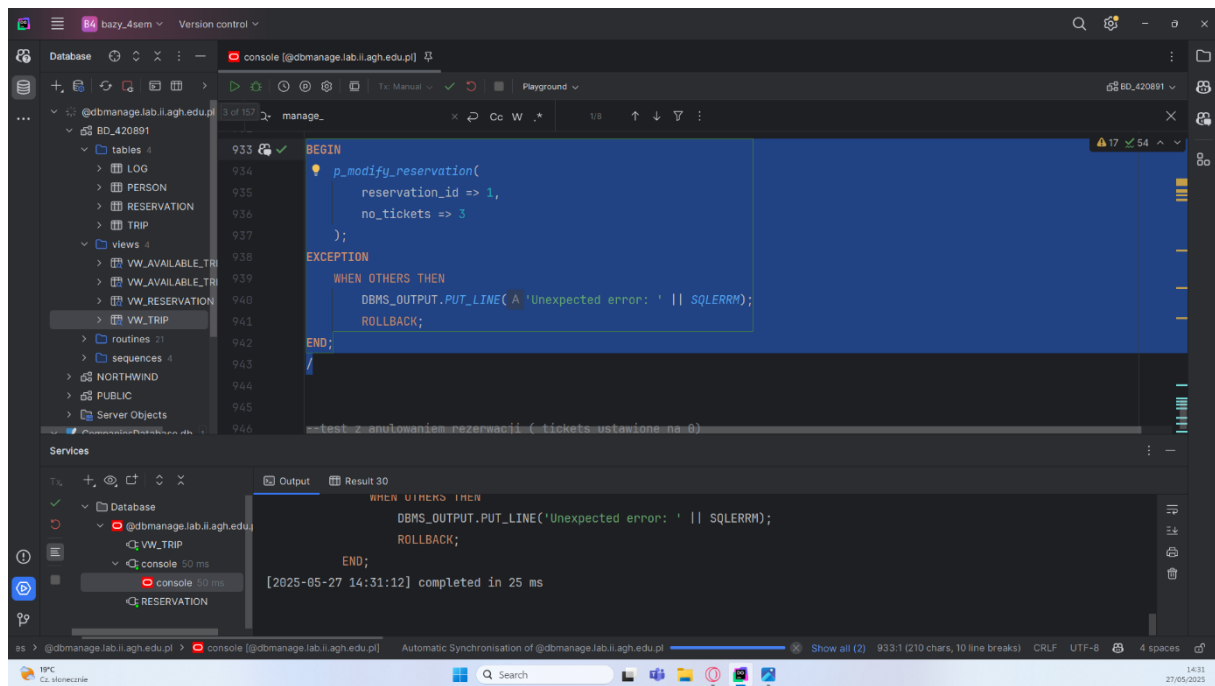
Output:

NO_TICKETS	NO_AVAILABLE_PLACES
1	1



test dla trg\_change\_no\_tickets:







test dla trg\_change\_no\_tickets:

The screenshot shows the SQL Developer interface with a query window open. The query is a SELECT statement that joins the reservation and trip tables, filtering for reservation\_id = 3. The results are displayed in a table with two columns: NO\_TICKETS and NO\_AVAILABLE\_PLACES. The output shows 4 tickets and 0 available places.

```
946 --test z anulowaniem rezerwacji ( tickets ustawione na 0)
947 SELECT r.no_tickets, t.no_available_places
948 FROM reservation r JOIN trip t ON r.trip_id = t.trip_id
949 WHERE r.reservation_id = 3;
950
951 BEGIN
952     p_modify_reservation(
953         reservation_id => 1,
954         no_tickets => 0
955     );
956 EXCEPTION
957 WHEN OTHERS THEN
958     DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
959 ROLLBACK;
```

NO_TICKETS	NO_AVAILABLE_PLACES
4	0

The screenshot shows the SQL Developer interface with a PL/SQL procedure call. The procedure p\_modify\_reservation is called with reservation\_id = 3 and no\_tickets = 0. The output window shows the execution details, including the completion time and the number of rows affected.

```
949 WHERE r.reservation_id = 3;
950
951 BEGIN
952     p_modify_reservation(
953         reservation_id => 3,
954         no_tickets => 0
955     );
956 EXCEPTION
957 WHEN OTHERS THEN
958     DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
959 ROLLBACK;
960 END;
```

Output:

```
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
ROLLBACK;
END;
[2025-05-27 14:32:25] completed in 19 ms
```

Database Explorer

console [dbmanage.lab.ii.agh.edu.pl] | RESERVATION [dbmanage.lab.ii.agh.edu.pl]

Q: manage\_

```
946 --test z anulowaniem rezerwacji ( tickets ustawione na 0)
947 SELECT r.no_tickets, t.no_available_places
948 FROM reservation r JOIN trip t ON r.trip_id = t.trip_id
949 WHERE r.reservation_id = 3;
950
951 BEGIN
952     p_modify_reservation(
953         reservation_id => 3,
954         no_tickets => 0
955     );
956 EXCEPTION
957     WHEN OTHERS THEN
958         DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
959     ROLLBACK;
```

Services

Output Result 38

NO_TICKETS	NO_AVAILABLE_PLACES
1	4

1 row

Database Consoles > @dbmanage.lab.ii.agh.edu.pl > console [dbmanage.lab.ii.agh.edu.pl]

947:1 (126 chars, 2 line breaks) CRLF UTF-8 4 spaces

1432 27/05/2023