

Laboratory work 9

1. A passenger cancels their booking. You need to remove the booking for the flight. Ensure the ‘booking’ table no longer contains the booking. Simulate an error to test rollback (for example, invalid booking_id).

The screenshot shows a PostgreSQL database interface with the following details:

- Database Explorer:** Shows the database structure. Under the `public` schema, there are 10 tables: `airline`, `airport`, `baggage`, `boarding`, `booking`, `column`, and `boc`.
- console_7 [postgres@localhost [2]]:** This tab contains a transaction log with the following SQL statements:

```
1 ✓ BEGIN;
2 ✓ DELETE FROM baggage_check WHERE booking_id = 12;
3 ✓ DELETE FROM baggage WHERE booking_id = 12;
4 ✓ DELETE FROM booking_flight WHERE booking_id = 12;
5 ✓ DELETE FROM booking WHERE booking_id=12;
6 ✓ SELECT * FROM booking WHERE booking_id=12;
7 --RAISE EXCEPTION 'Simulated error: cancel booking failed';
8 ✓ COMMIT ;
```
- Database Sessions:** Shows two sessions: `postgres@localhost` and `console`. The `postgres@localhost` session is active.
- Output:** Shows the results of a query on the `booking` table, which has 0 rows.

2. Rescheduling a flight. You need to reschedule a flight. Verify the ‘flights’ table reflects the new departure time. Simulate an error to test rollback (for example, invalid flight_id)

The screenshot shows a PostgreSQL database console interface. In the top right, there's a tab labeled "console_7 [postgres@localhost [2]]". The main area contains a code editor with the following SQL script:

```

ROLLBACK;

BEGIN;
UPDATE flights
SET sch_departure_time = '2025-12-12 10:00:00',
    sch_arrival_time = '2025-12-12 13:00:00',
    updated_at=now()
WHERE flight_id = 6;

SELECT flight_id,sch_departure_time, sch_arrival_time FROM flights
WHERE flight_id=6;

-- RAISE EXCEPTION 'Simulated error: reschedule flight';
COMMIT ;

```

Below the code editor is a "Database Sessions" panel showing a single session with one transaction. The transaction details are as follows:

flight_id	sch_departure_time	sch_arrival_time
6	2025-12-12 10:00:00.000000	2025-12-12 13:00:00.000000

At the bottom of the interface, it says "Database Consoles > postgres@localhost [2] > console_7 [postgres@localhost [2]]".

3. Updating ticket prices. You need to decrease the ticket price for a specific flight for all existing bookings. If an error occurs, no changes should be applied.

The screenshot shows a PostgreSQL database console interface. In the top right, there's a tab labeled "console_7 [postgres@localhost [2]]". The main area contains a code editor with the following SQL script:

```

BEGIN;
UPDATE booking
SET ticket_price = ticket_price - 10,
    updated_at = NOW()
WHERE flight_id = 5;

SELECT booking_id, ticket_price
FROM booking
WHERE flight_id = 5;
-- RAISE EXCEPTION 'Simulated error reducing price';
COMMIT;

```

Below the code editor is a "Database Sessions" panel showing a single session with one transaction. The transaction details are as follows:

booking_id	ticket_price
1	340.50
16	540.25
24	300.00
36	540.25
44	310.00
56	545.00
96	555.00
104	315.00
116	560.00
124	325.00
136	565.00

At the bottom of the interface, it says "Database Consoles > postgres@localhost [2] > console_7 [postgres@localhost [2]]".

4. A passenger updates their details. Ensure the update is reflected across all associated records, including bookings.

The screenshot shows a PostgreSQL database interface with the following details:

- Database Explorer:** Shows the database structure with the `passenger_id` column highlighted in the `passengers` table under the `airline` schema.
- Playground:** Contains the following SQL code:


```

1 ✓ BEGIN;
2 ✓ UPDATE passengers
  SET first_name = 'Aslan',
      last_name = 'Sultan',
      country_of_residence = 'Kazakhstan',
      updated_at = NOW()
  WHERE passenger_id = 89;
3 ✓
4 ✓
5 ✓
6 ✓
7 ✓
8 ✓
9 ✓ SELECT * FROM passengers WHERE passenger_id = 89;
10 ✓ COMMIT;
      
```
- Database Sessions:** Shows the result of the update query in the `passengers` table:

passenger_id	first_name	last_name	date_of_birth	gender	country_of_citizenship
89	Aslan	Sultan	1983-07-17	Male	Italy

5. A new passenger is registered, and a booking is created. Ensure the new passenger is added and the booking succeeds.

The screenshot shows a PostgreSQL database interface with the following details:

- Database Explorer:** Shows the database structure with the `passenger_id` column highlighted in the `passengers` table under the `airline` schema.
- Playground:** Contains the following SQL code:


```

1 ✓ BEGIN;
2 ✓ INSERT INTO passengers (
  passenger_id, first_name, last_name, date_of_birth,
  gender, country_of_citizenship, country_of_residence,
  passport_number, created_at, updated_at
)
VALUES (
  passenger_id 999, first_name 'Aruzhan', last_name 'Samat',
  date_of_birth '2003-08-15', gender 'Female',
  country_of_citizenship 'Kazakhstan', country_of_residence 'Kazakhstan',
  passport_number 'KZ1234567',
  created_at NOW(), updated_at NOW()
);
3 ✓
4 ✓
5 ✓
6 ✓
7 ✓
8 ✓
9 ✓
10 ✓
11 ✓
12 ✓
13 ✓
14 ✓
15 ✓
16 ✓
17 ✓
18 ✓
19 ✓
20 ✓
21 ✓
22 ✓
23 ✓
24 ✓
25 ✓
26 ✓
      
```
- Database Sessions:** Shows the results of the insertions:

passenger_id	first_name	last_name	date_of_birth	gender	country_of_citizenship
999	Aruzhan	Samat	2003-08-15	Female	Kazakhstan

Database Explorer

```

14 ✓ INSERT INTO booking (
15   booking_id, flight_id, passenger_id, booking_platform,
16   created_at, updated_at, status, ticket_price
17 )
18 VALUES (
19   booking_id 888, flight_id 5, passenger_id 999, booking_platform 'Web App',
20   created_at NOW(), updated_at NOW(), status 'Confirmed', ticket_price 130.00
21 );
22 ✓ SELECT * FROM passengers WHERE passenger_id = 999.

```

Database Sessions

	booking_id	flight_id	passenger_id	booking_platform	created_at	updated_at
postgres@1	888	5	999	Web App	2025-11-25 21:36:58.350713	2025-11-25 21:36:58.350713

Database Consoles > postgres@localhost [2] > console_7 [postgres@localhost [2]]

Database Explorer

```

14 ✓ INSERT INTO booking (
15   booking_id, flight_id, passenger_id, booking_platform,
16   created_at, updated_at, status, ticket_price
17 )
18 VALUES (
19   booking_id 888, flight_id 5, passenger_id 999, booking_platform 'Web App',
20   created_at NOW(), updated_at NOW(), status 'Confirmed', ticket_price 130.00
21 );
22 ✓ SELECT * FROM passengers WHERE passenger_id = 999.

```

Database Sessions

	passenger_id	first_name	last_name	date_of_birth	gender	country_of_citizenship
postgres@1	999	Aruzhan	Samat	2003-08-15	Female	Kazakhstan

Database Consoles > postgres@localhost [2] > console_7 [postgres@localhost [2]]

6. Increase the ticket price for all bookings on a specific flight by a fixed amount.

The screenshot shows a PostgreSQL database client interface. In the top right, there's a tab labeled "console_7 [postgres@localhost [2]]". Below it, the "Database Explorer" pane shows a connection to "localhost [2]". The main area displays a SQL script:

```

1 ✓ BEGIN;
2 ✓ UPDATE booking
3     SET ticket_price = ticket_price + 15,
4         updated_at = NOW()
5     WHERE flight_id = 5;
6 ✓ SELECT booking_id, ticket_price FROM booking WHERE flight_id = 5;
7 ✓ COMMIT;

```

Below the script, the "Database Sessions" pane shows a table named "booking" with 21 rows. The columns are "booking_id" and "ticket_price". The data is as follows:

booking_id	ticket_price
1	355.50
16	555.25
24	315.00
36	555.25
44	325.00
56	560.00
96	570.00
104	330.00
116	575.00
124	340.00
136	580.00
64	330.00
76	565.00
13	
5	
6	
7	
8	
9	
10	
11	
12	
14	
15	

At the bottom, the status bar shows "7:8 LF UTF-8 4 spaces ⌂ ⌂".

7. Update a baggage weight. A passenger updates the declared weight of their baggage. Ensure that the change is correctly reflected in the database.

The screenshot shows a PostgreSQL database client interface. In the top right, there's a tab labeled "console_7 [postgres@localhost [2]]". Below it, the "Database Explorer" pane shows a connection to "localhost [2]". The main area displays a SQL script:

```

1 ✓ BEGIN;
2 ✓ UPDATE baggage
3     SET weight_in_kg = 15.50,
4         updated_at = NOW()
5     WHERE baggage_id = 3;
6 ✓ SELECT * FROM baggage WHERE baggage_id = 3;
7 ✓ COMMIT;

```

Below the script, the "Database Sessions" pane shows a table named "baggage" with 1 row. The columns are "baggage_id", "weight_in_kg", "created_at", "updated_at", and "booking_id". The data is as follows:

baggage_id	weight_in_kg	created_at	updated_at	booking_id
3	15.50	2025-09-30 14:14:46.168315	2025-11-25 21:40:14.233796	25

At the bottom, the status bar shows "1:7 LF UTF-8 4 spaces ⌂ ⌂".

8. Apply a discount to a booking for a specific passenger. If any error occurs,

roll back.

The screenshot shows a PostgreSQL database console interface. The code in the main window is:

```
1 ✓ BEGIN;
2 ✓ UPDATE booking
3     SET ticket_price = ticket_price * 0.85,
4         updated_at = NOW()
5     WHERE booking_id = 8;
6
7 ✓ SELECT booking_id, ticket_price FROM booking
8     WHERE booking_id = 8;
9     -- RAISE EXCEPTION 'Discount error: rollback required';
10 ✓ COMMIT;
```

The output window shows a single row from the 'booking' table:

booking_id	ticket_price
8	246.50

The status bar at the bottom indicates the transaction was rolled back.

9. Reschedule all bookings for a flight to a new flight.

The screenshot shows a PostgreSQL database console interface. The code in the main window is:

```
1 ✓ BEGIN;
2 ✓ UPDATE booking
3     SET flight_id = 6,
4         updated_at = NOW()
5     WHERE flight_id = 5;
6
7 ✓ SELECT booking_id, flight_id FROM booking
8     WHERE flight_id IN (5,6);
9
10 -- RAISE EXCEPTION 'Reschedule failure: rollback';
11 ✓ COMMIT;
```

The output window shows multiple rows from the 'booking' table, indicating a failure to update the 'flight_id' column for some bookings:

booking_id	flight_id
1	9
2	29
3	49
4	69
5	89
6	1
7	16
8	24
9	36

The status bar at the bottom indicates the transaction was rolled back.