

## LAB 9. Abdykamat Adilet. TRANSACTION.

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).

```
BEGIN;
```

```
DELETE FROM baggage_check
```

```
WHERE booking_id = 10;
```

```
DELETE FROM baggage
```

```
WHERE booking_id = 10;
```

```
DELETE FROM booking
```

```
WHERE booking_id = 10;
```

```
DELETE FROM booking
```

```
WHERE booking_id = -1;
```

```
ROLLBACK;
```

The screenshot displays a SQL IDE interface with two main panels. The top panel, titled 'Query' and 'Query History', shows a SQL script with 14 lines of code. The script begins with 'BEGIN;', followed by three 'DELETE' statements targeting 'baggage\_check', 'baggage', and 'booking' tables, each with a 'WHERE' clause using 'booking\_id'. The first two deletions use 'booking\_id = 10', while the third uses 'booking\_id = -1'. The script concludes with 'ROLLBACK;'. The bottom panel, titled 'Data Output', 'Messages', and 'Notifications', shows the execution result: 'ROLLBACK' followed by 'Query returned successfully in 98 msec.'. A green status bar at the bottom right confirms 'Query returned successfully in 98 msec.' with a green checkmark icon.

```
1 BEGIN;  
2 DELETE FROM baggage_check  
3 WHERE booking_id = 10;  
4  
5 DELETE FROM baggage  
6 WHERE booking_id = 10;  
7  
8 DELETE FROM booking  
9 WHERE booking_id = 10;  
10  
11 DELETE FROM booking  
12 WHERE booking_id = -1;  
13  
14 ROLLBACK;
```

ROLLBACK

Query returned successfully in 98 msec.

✓ Query returned successfully in 98 msec. ✕

**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).**

```
BEGIN;

UPDATE flights

SET sch_departure_time = '2025-01-30 14:00:00'

WHERE flight_id = 5;

-- Error simulation

UPDATE flights

SET sch_departure_time = '2025-01-30 15:00:00'

WHERE flight_id = -1;

ROLLBACK;
```

The screenshot shows a SQL IDE interface. The top pane, titled 'Query', contains the following SQL script:

```
1 BEGIN;
2
3 UPDATE flights
4 SET sch_departure_time = '2025-01-30 14:00:00'
5 WHERE flight_id = 5;
6
7 -- Error simulation
8 UPDATE flights
9 SET sch_departure_time = '2025-01-30 15:00:00'
10 WHERE flight_id = -1;
11
12 ROLLBACK;
13
```

The bottom pane has tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is active and displays the text 'ROLLBACK' and 'Query returned successfully in 97 msec.'.

A green status bar at the bottom right of the IDE shows a checkmark icon and the text 'Query returned successfully in 97 msec.'.

**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.**

```
BEGIN;
```

```
UPDATE tickets
```

```
SET price = price - 50
```

```
WHERE flight_number = 'SU1002';
```

```
-- Error simulation
```

```
UPDATE tickets
```

```
SET price = price / 0;
```

```
ROLLBACK;
```



The screenshot shows a SQL IDE interface with a query editor and a messages panel. The query editor contains the following SQL code:

```
1 BEGIN;  
2  
3 UPDATE tickets  
4 SET price = price - 50  
5 WHERE flight_number = 'SU1002';  
6  
7 -- Error simulation  
8 UPDATE tickets  
9 SET price = price / 0;  
10  
11 ROLLBACK;
```

The messages panel at the bottom shows an error message:

ERROR: division by zero

SQL state: 22012

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

```
BEGIN;

UPDATE passengers

SET first_name = 'John',

    last_name = 'Williams',

    country_of_citizenship = 'USA'

WHERE passenger_id = 12;

UPDATE booking

SET passenger_id = 12

WHERE passenger_id = 12;

-- Error simulation

UPDATE passengers

SET first_name = NULL

WHERE passenger_id = -1;

ROLLBACK;
```

The screenshot displays a SQL IDE interface. The top pane, titled 'Query' and 'Query History', contains the following SQL code:

```
4 SET first_name = 'John';
5     last_name = 'Williams';
6     country_of_citizenship = 'USA'
7 WHERE passenger_id = 12;
8
9 UPDATE booking
10 SET passenger_id = 12
11 WHERE passenger_id = 12;
12
13 -- Error simulation
14 UPDATE passengers
15 SET first_name = NULL
16 WHERE passenger_id = -1;
17
18 ROLLBACK;
```

The bottom pane is divided into 'Data Output', 'Messages', and 'Notifications' tabs. The 'Messages' tab is active and shows the text 'ROLLBACK' followed by 'Query returned successfully in 97 msec.' A green status bar at the bottom right of the IDE confirms: '✓ Query returned successfully in 97 msec. ✕'.

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

BEGIN;

INSERT INTO passengers (first\_name, last\_name, country\_of\_citizenship, date\_of\_birth,  
passport\_number,

gender, passenger\_id, country\_of\_residence, created\_at,  
updated\_at )

VALUES ('Adam', 'Stone', 'Canada', '1990-01-01', 'AB1234567', 'M', '99', 'India', NOW(), NOW())

RETURNING passenger\_id;

INSERT INTO booking ( booking\_id, passenger\_id, booking\_platform, flight\_id, created\_at, updated\_at,  
status, ticket\_price, ticket\_discount)

VALUES ( 1001, 5, 'Mobile App', 2003, NOW(), NOW(), 'Co', 0, 0 );

ROLLBACK;

COMMIT;



The screenshot shows a database query editor with a 'Query' tab selected. The SQL code is as follows:

```
20 VALUES (
21     1001,
22     5,
23     'Mobile App',
24     2003,
25     NOW(),
26     NOW(),
27     'Co',
28     0,
29     0
30 );
31
32 ROLLBACK;
33
34 COMMIT;
```

Below the code editor, there are three tabs: 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is active, displaying the following warning:

WARNING: there is no transaction in progress  
COMMIT

Below the warning, it states: 'Query returned successfully in 86 msec.'

In the bottom right corner, there is a green status bar with a checkmark icon and the text: 'Query returned successfully in 86 msec. X'

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

BEGIN;

UPDATE tickets

SET price = price + 30

WHERE flight\_number = 'SU1004';

ROLLBACK;

COMMIT;

Query

Query History

1

BEGIN;

2

3

▼ UPDATE tickets

4 SET price = price + 30

5 WHERE flight\_number = 'SU1004';

6 ROLLBACK;

7 COMMIT;

Data Output

Messages

Notifications

WARNING: there is no transaction in progress

COMMIT

Query returned successfully in 93 msec.

✓ Query returned successfully in 93 msec.

✕

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

```
BEGIN;
```

```
UPDATE baggage
```

```
SET weight_in_kg = 23.5
```

```
WHERE baggage_id = 77;
```

```
-- Error simulation
```

```
UPDATE baggage
```

```
SET weight_in_kg = -5
```

```
WHERE baggage_id = 77;
```

```
ROLLBACK;
```

```
-- COMMIT;
```

The screenshot displays a SQL query editor interface. The top section, titled 'Query', shows a multi-line SQL script. The script begins with 'BEGIN;', followed by an 'UPDATE baggage' statement that sets 'weight\_in\_kg' to 23.5 for 'baggage\_id' 77. A comment '-- Error simulation' precedes a second 'UPDATE baggage' statement that sets 'weight\_in\_kg' to -5 for the same 'baggage\_id'. This is followed by 'ROLLBACK;' and a final comment '-- COMMIT;'. The bottom section, titled 'Messages', shows the output of the 'ROLLBACK' command, stating 'Query returned successfully in 97 msec.' A green status bar at the bottom right of the interface confirms this, displaying a checkmark and the text 'Query returned successfully in 97 msec.'.

```
1 BEGIN;  
2  
3 UPDATE baggage  
4 SET weight_in_kg = 23.5  
5 WHERE baggage_id = 77;  
6  
7 -- Error simulation  
8 UPDATE baggage  
9 SET weight_in_kg = -5  
10 WHERE baggage_id = 77;  
11  
12 ROLLBACK;  
13 -- COMMIT;
```

ROLLBACK

Query returned successfully in 97 msec.

✓ Query returned successfully in 97 msec. ✕

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

```
BEGIN;
```

```
UPDATE booking
```

```
SET ticket_price = ticket_price * 0.85
```

```
WHERE booking_id = 300;
```

```
-- Error simulation
```

```
UPDATE booking
```

```
SET ticket_price = ticket_price / 0;
```

```
ROLLBACK;
```

```
-- COMMIT;
```



The screenshot shows a SQL IDE interface with a query editor and a messages pane. The query editor contains the following SQL code:

```
1 BEGIN;  
2  
3 UPDATE booking  
4 SET ticket_price = ticket_price * 0.85  
5 WHERE booking_id = 300;  
6  
7 -- Error simulation  
8 UPDATE booking  
9 SET ticket_price = ticket_price / 0;  
10  
11 ROLLBACK;  
12 -- COMMIT;
```

The messages pane at the bottom shows the following error message:

```
ERROR: division by zero  
  
SQL state: 22012
```



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

```
BEGIN;
```

```
UPDATE tickets
```

```
SET flight_number = 'SU1001'
```

```
WHERE flight_number = 'SU1005';
```

```
-- Error simulation
```

```
UPDATE flights
```

```
SET sch_departure_time = NULL
```

```
WHERE flight_id = -1;
```

```
ROLLBACK;
```

```
-- COMMIT;
```

The screenshot shows a SQL IDE interface. The top pane, titled 'Query', contains a SQL script with 13 lines. Lines 1-5 are expanded, showing: `1 BEGIN;`, `2`, `3 UPDATE tickets`, `4 SET flight_number = 'SU1001'`, and `5 WHERE flight_number = 'SU1005';`. Lines 6-13 are collapsed. The bottom pane, titled 'Messages', shows a 'ROLLBACK' message and a success message: 'Query returned successfully in 125 msec.'.

```
1 BEGIN;
2
3 UPDATE tickets
4 SET flight_number = 'SU1001'
5 WHERE flight_number = 'SU1005';
6
7 -- Error simulation
8 UPDATE flights
9 SET sch_departure_time = NULL
10 WHERE flight_id = -1;
11
12 ROLLBACK;
13 -- COMMIT;
```

ROLLBACK

Query returned successfully in 125 msec.

✓ Query returned successfully in 125 msec. ✕