

## Lab 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 pgAdmin interface with a database console titled "console [db\_international\_airports]". The script pane contains the following code:

```
695  
696  
697  
698  
699  
700  
701 begin;  
702 delete from booking  
703 where booking_id = 25;  
704  
705 ✓ select *from booking where booking_id = 25;  
706  
707
```

The output pane shows the results of the last query:

booking_id	flight_id	passenger_id	booking_platform	created_at	updated_at	status	ticket_price

Services and Tx tabs are visible at the bottom.

The screenshot shows a pgAdmin interface with a database console titled "console [db\_international\_airports]". The script pane contains the following code:

```
705  
706  
707  
708  
709  
710 d select *from booking where booking_id = 25;  
711  
712 begin;  
713  
714 delete from booking  
715 where booking_id = 99999;  
716 select* from booking where booking_id = 99999;  
717  
718 ✓ select* from booking limit 25;  
719
```

The output pane shows the results of the last query:

booking_id	flight_id	passenger_id	booking_platform	created_at	updated_at	status	ticket
18	18	7	169 Rank	2025-09-24 06:10:42.755116	2025-09-24 06:10:42.755116	checked in	
19	19	81	86 Rank	2025-09-24 06:10:42.766519	2025-09-24 06:10:42.766519	confirmed	
20	20	73	191 Tresom	2025-09-24 06:10:42.775914	2025-09-24 06:10:42.775914	confirmed	
21	21	158	25 Andalax	2025-09-24 06:10:42.784734	2025-09-24 06:10:42.784734	checked in	
22	22	93	154 Prodder	2025-09-24 06:10:42.792889	2025-09-24 06:10:42.792889	boarded	
23	23	105	17 Alphazap	2025-09-24 06:10:42.803192	2025-09-24 06:10:42.803192	cancelled	
24	24	89	103 Trippledex	2025-09-24 06:10:42.813582	2025-09-24 06:10:42.813582	cancelled	
25	25	198	55 Mat Lam Tam	2025-09-24 06:10:42.824373	2025-09-24 06:10:42.824373	no show	

Services and Tx tabs are visible at the bottom.

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 interface with a transaction block open. The transaction code is as follows:

```

719
720 begin;
721 update flights
722 set sch_departure_time = '2025-02-15 14:30:00'
723 where flight_id = 10;
724
725 commit;
726
727 begin;
728 update flights
729 set sch_departure_time = '2025-02-20 09:00:00'
730 where flight_id = 99999;
731
732 select* from flights where flight_id = 99999;
733 select 1/0;
734 rollback;
735
736 select flight_id, sch_departure_time from flights where flight_id = 10;
    
```

The transaction ends with a successful result:

```

738 ✓ select flight_id, sch_departure_time from flights where flight_id = 10;

```

The Services tab shows the output of the query:

Flight_id	sch_departure_time
10	2025-02-15 14:30:00.000000

Database Consoles > db\_international\_airports > console [db\_international\_airports]

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 interface with a transaction block open. The transaction code is as follows:

```

740
741 begin;
742 update booking
743 set ticket_price = ticket_price*0.8
744 where flight_id = 105;
745
746 commit;
747
748 begin;
749 update booking
750 set ticket_price = ticket_price * 0.8
751 where flight_id = 105;
752
753 update booking
754 set no_such_column = 123;
755 rollback;
756
757
758 ✓ select flight_id, ticket_price from booking where flight_id = 105;
    
```

The transaction ends with a successful result:

```

759

```

The Services tab shows the output of the query:

flight_id	ticket_price	
1	105	19.19
2	105	25.59
3	105	25.59
4	105	1.91
5	105	28.79
6	105	19.19

Database Consoles > db\_international\_airports > console [db\_international\_airports]

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 `db_international_airports` database with the `passenger` table selected. The table has columns: `passenger_id`, `first_name`, `last_name`, `date_of_birth`, `gender`, `country_of_citizenship`, `country_of_residence`, and `passport_number`.
- Console:** Displays the following SQL code:
 

```

begin;
update passengers
set first_name = 'New',
    last_name = 'Name',
    passport_number = '985-974-98-23'
where passenger_id = 77;
commit;

begin;
update passengers
set passport_number = 'Invalid number'
where passenger_id = 77;
rollback;

select * from passengers where passenger_id = 77;
      
```
- Services - Output:** Shows the result of the final `select` statement:
 

passenger_id	first_name	last_name	date_of_birth	gender	country_of_citizenship	country_of_residence	passport_number
77	New	Name	2025-06-25	Female	Brazil	Philippines	985-974-98-23

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 `db_international_airports` database with the `passenger` and `booking` tables selected.
- Console:** Displays the following SQL code:
 

```

begin;
insert into passengers(passenger_id, first_name, last_name, date_of_birth, gender, country_of_citizenship, country_of_residence, passport_number)
values ( passenger_id 498, first_name 'Aruzhan', last_name 'Sadykova', date_of_birth '2005-08-25', gender 'Female', country_of_citizenship 'Kazakhstan', country_of_residence 'Kazakhstan', passport_number '985-974-98-23' );

insert into booking(booking_id, flight_id, passenger_id, booking_platform, status, ticket_price)
values( booking_id 402, flight_id 10, passenger_id 498, booking_platform 'Aviata', status 'BOARDED', ticket_price 657);

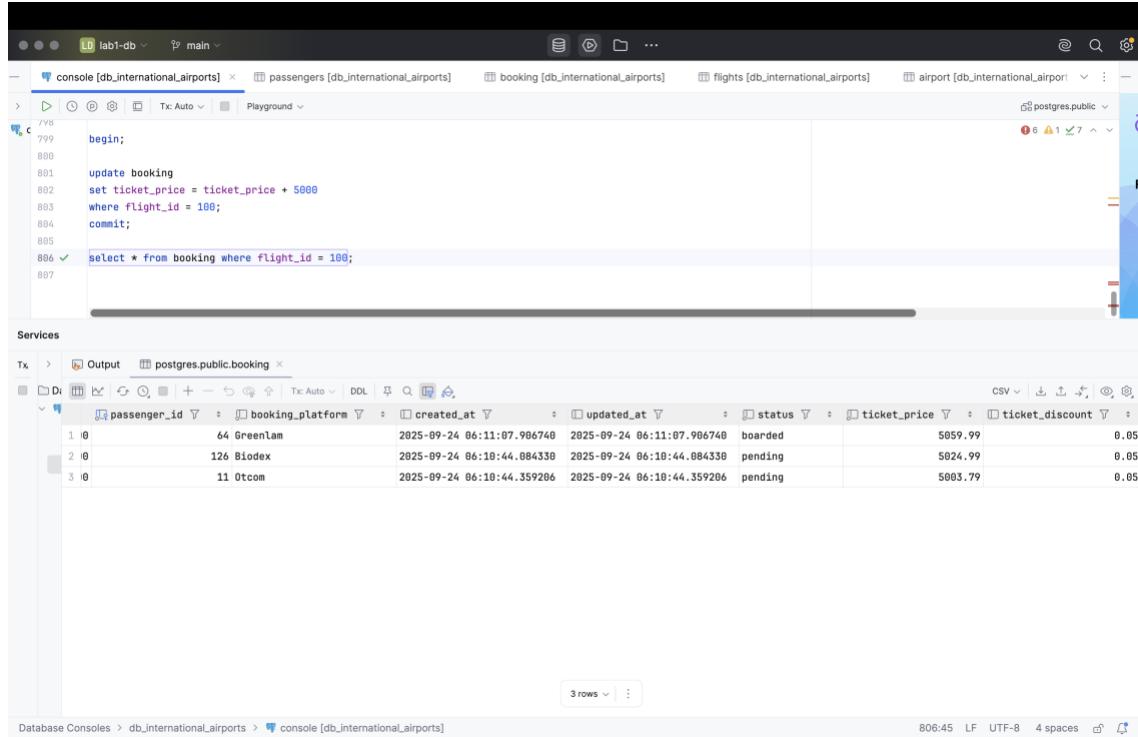
commit;

begin;
insert into passengers(passenger_id, first_name, last_name, date_of_birth, gender, country_of_citizenship, country_of_residence, passport_number)
values ( passenger_id 499, first_name 'Test', last_name 'Error', date_of_birth '2008-08-03', gender 'Male', country_of_citizenship 'Russia', country_of_residence 'Russia', passport_number '985-974-98-23' );
insert into booking(booking_id, flight_id, passenger_id, booking_platform, status, ticket_price)
values( booking_id 403, flight_id 398, passenger_id 499, booking_platform 'Aviata', status 'Boarded', ticket_price 75);
rollback;

select * from booking where passenger_id = 498;
      
```
- Services - Output:** Shows the result of the final `select` statement:
 

booking_id	flight_id	passenger_id	booking_platform	status	ticket_price
402	10	498	Aviata	BOARDED	657

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



```

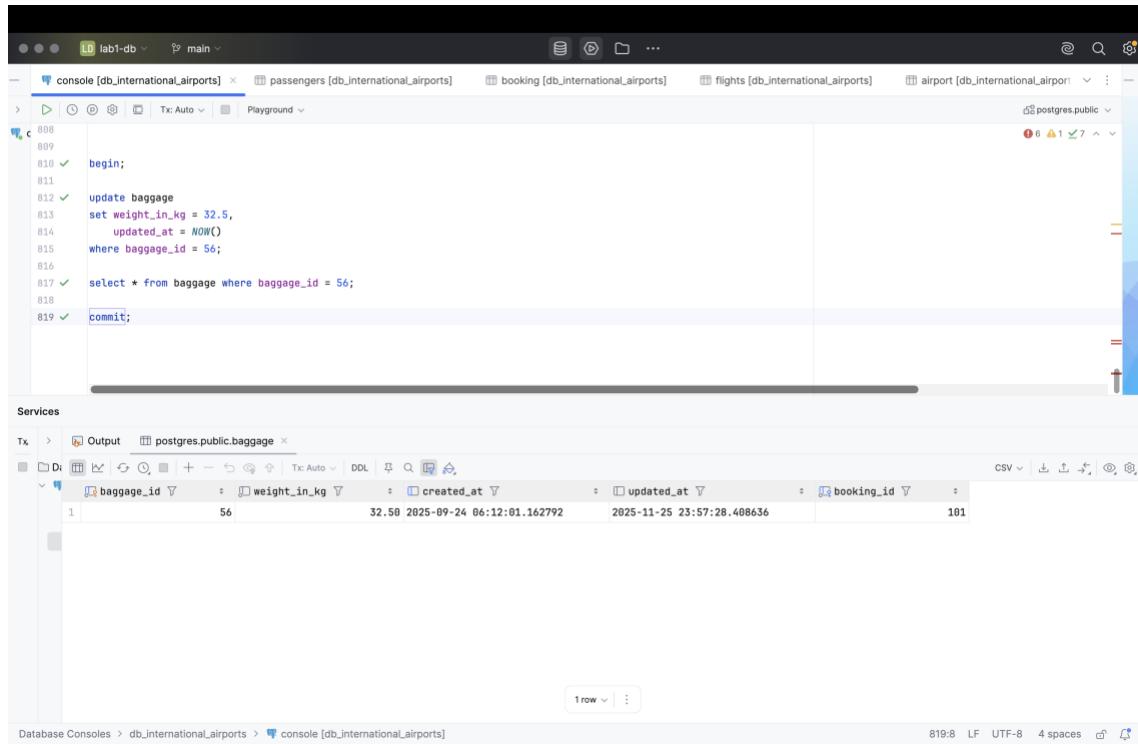
799 begin;
800
801 update booking
802 set ticket_price = ticket_price + 5000
803 where flight_id = 100;
804 commit;
805
806 select * from booking where flight_id = 100;
807

```

The screenshot shows a PostgreSQL database console with a transaction block. The transaction begins with a `begin;` statement. It then performs an `update` operation on the `booking` table, setting the `ticket_price` column to its current value plus 5000, specifically for rows where `flight_id` is 100. After the update, a `commit;` statement is executed. Finally, a `select *` statement is run to verify the changes, filtering for `flight_id = 100`. The results show three rows of data with updated ticket prices.

passenger_id	booking_platform	created_at	updated_at	status	ticket_price	ticket_discount
1	64 Greenlam	2025-09-24 06:11:07.906740	2025-09-24 06:11:07.906740	boarded	5059.99	0.05
2	126 Biodex	2025-09-24 06:10:44.084330	2025-09-24 06:10:44.084330	pending	5024.99	0.05
3	11 Otcom	2025-09-24 06:10:44.359266	2025-09-24 06:10:44.359266	pending	5003.79	0.05

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



```

808
809
810 begin;
811
812 update baggage
813 set weight_in_kg = 32.5,
814     updated_at = NOW()
815 where baggage_id = 56;
816
817 select * from baggage where baggage_id = 56;
818
819 commit;

```

The screenshot shows a PostgreSQL database console with a transaction block. The transaction begins with a `begin;` statement. It then performs an `update` operation on the `baggage` table, setting the `weight_in_kg` column to 32.5 and the `updated_at` column to the current timestamp (`NOW()`), specifically for the row where `baggage_id` is 56. After the update, a `select *` statement is run to verify the changes, filtering for `baggage_id = 56`. The results show one row of data with the updated weight.

baggage_id	weight_in_kg	created_at	updated_at	booking_id
56	32.50	2025-09-24 06:12:01.162792	2025-11-25 23:57:28.408636	181

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

The screenshot shows a PostgreSQL database interface with a transaction log and a table output.

**Transaction Log:**

```

823
824 update booking
825 set ticket_discount = 0.1,
826      updated_at = NOW()
827 where passenger_id = 77;
828
829 rollback;
830
831 ✓ select * from booking where passenger_id = 77;
832
833
834
835

```

**Table Output:**

ht_id	passenger_id	booking_platform	created_at	updated_at	status	ticket_price	ticket_discount
1	188	77 Dalfresh	2025-09-24 06:11:00.086526	2025-09-24 06:11:00.086526	cancelled	29.99	
2	256	77 Solarbreeze	2025-09-30 23:51:36.690537	2025-09-30 23:51:36.690537	BOARDED	18.99	

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

The screenshot shows a PostgreSQL database interface with a transaction log and a table output.

**Transaction Log:**

```

832
833 begin;
834
835 update booking
836 set flight_id = 101,
837      updated_at = now()
838 where flight_id = 103;
839
840 ✓ select * from booking where flight_id = 101;
841
842
843
844

```

**Table Output:**

booking_id	flight_id	passenger_id	booking_platform	created_at	updated_at	status	ticket_price
1	354	101	368 Wrapsafe	2025-09-30 23:51:36.763537	2025-11-26 00:06:32.659129	PENDING	