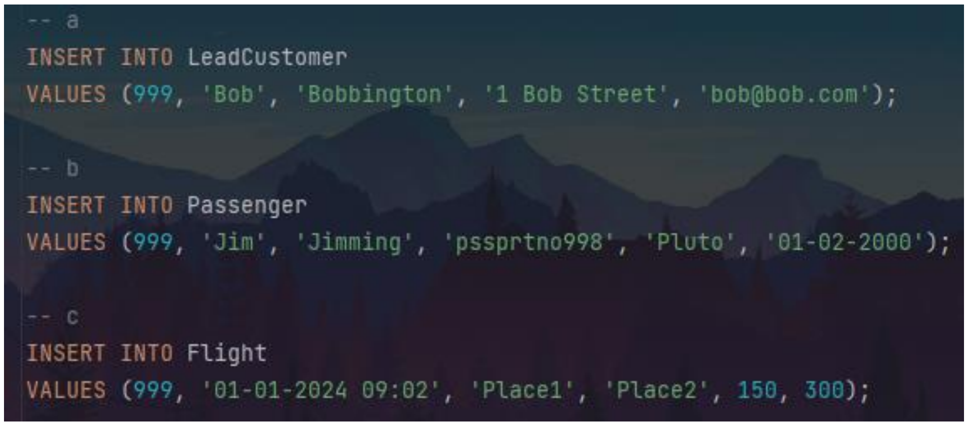
Transactions of Interest

A. Insert a new record. This could be

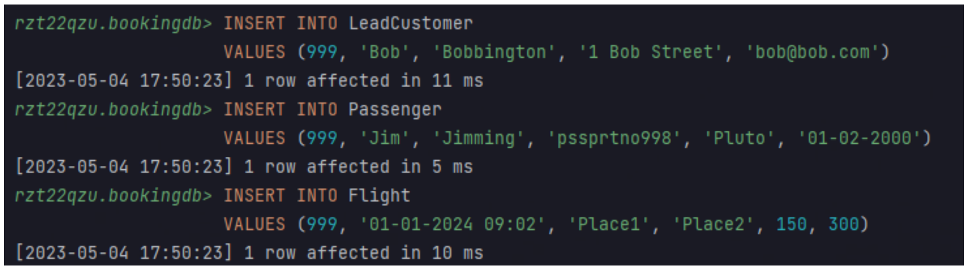
> **a. Given a lead customer ID number, name, and contact details, create a new customer record.**
> **b. Given a passenger with an ID, name, date of birth, etc., create a new passenger record.**
> **c. Given a flight ID number, origin, destination, flight date, capacity of the aircraft, and price per seat create a new flight record.**

SQL:



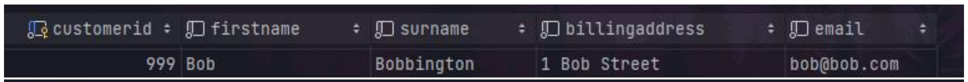
```
-- a
INSERT INTO LeadCustomer
VALUES (999, 'Bob', 'Bobbington', '1 Bob Street', 'bob@bob.com');

-- b
INSERT INTO Passenger
VALUES (999, 'Jim', 'Jimming', 'pssprtno998', 'Pluto', '01-02-2000');

-- c
INSERT INTO Flight
VALUES (999, '01-01-2024 09:02', 'Place1', 'Place2', 150, 300);
```

Output:

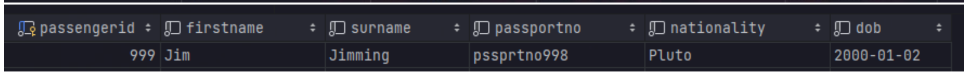


```
rzt22qzu.bookingdb> INSERT INTO LeadCustomer
                    VALUES (999, 'Bob', 'Bobbington', '1 Bob Street', 'bob@bob.com')
[2023-05-04 17:50:23] 1 row affected in 11 ms
rzt22qzu.bookingdb> INSERT INTO Passenger
                    VALUES (999, 'Jim', 'Jimming', 'pssprtno998', 'Pluto', '01-02-2000')
[2023-05-04 17:50:23] 1 row affected in 5 ms
rzt22qzu.bookingdb> INSERT INTO Flight
                    VALUES (999, '01-01-2024 09:02', 'Place1', 'Place2', 150, 300)
[2023-05-04 17:50:23] 1 row affected in 10 ms
```

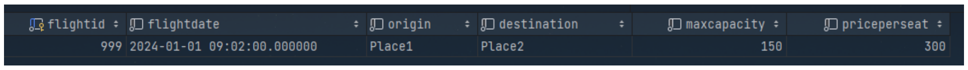
Result:



customerid	firstname	surname	billingaddress	email
999	Bob	Bobbington	1 Bob Street	bob@bob.com

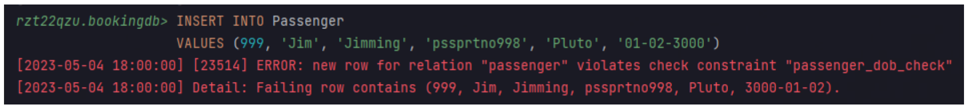


passengerid	firstname	surname	passportno	nationality	dob
999	Jim	Jimming	pssprtno998	Pluto	2000-01-02



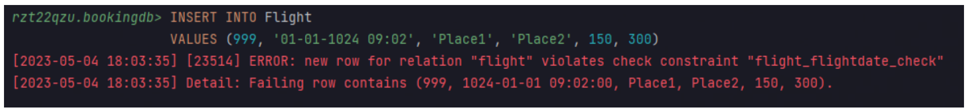
flightid	flightdate	origin	destination	maxcapacity	priceperseat
999	2024-01-01 09:02:00.000000	Place1	Place2	150	300

Erroneous testing: DoB in the future:



```
rzt22qzu.bookingdb> INSERT INTO Passenger
                    VALUES (999, 'Jim', 'Jimming', 'pssprtno998', 'Pluto', '01-02-3000')
[2023-05-04 18:00:00] [23514] ERROR: new row for relation "passenger" violates check constraint "passenger_dob_check"
[2023-05-04 18:00:00] Detail: Failing row contains (999, Jim, Jimming, pssprtno998, Pluto, 3000-01-02).
```

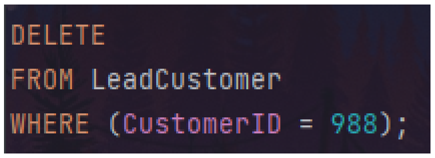
Erroneous testing: Flight in the past:



```
rzt22qzu.bookingdb> INSERT INTO Flight
                    VALUES (999, '01-01-1024 09:02', 'Place1', 'Place2', 150, 300)
[2023-05-04 18:03:35] [23514] ERROR: new row for relation "flight" violates check constraint "flight_flightdate_check"
[2023-05-04 18:03:35] Detail: Failing row contains (999, 1024-01-01 09:02:00, Place1, Place2, 150, 300).
```

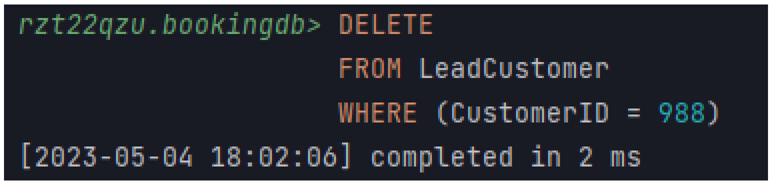
B. Given a customer ID number, remove the record for that customer. It should not be possible to remove customers that have active (i.e., reserved) flight bookings. A customer that has only cancelled bookings could be removed; the associated bookings should also be removed along with all the seat bookings.

SQL:



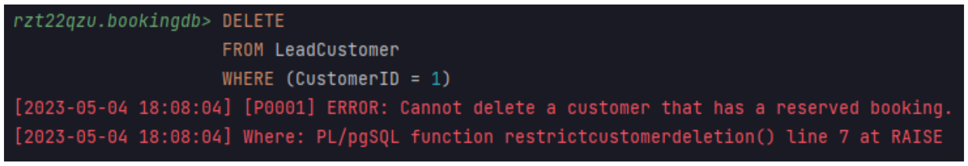
```
DELETE
FROM LeadCustomer
WHERE (CustomerID = 988);
```

Output:



```
rzt22qzu.bookingdb> DELETE
                    FROM LeadCustomer
                    WHERE (CustomerID = 988)
[2023-05-04 18:02:06] completed in 2 ms
```

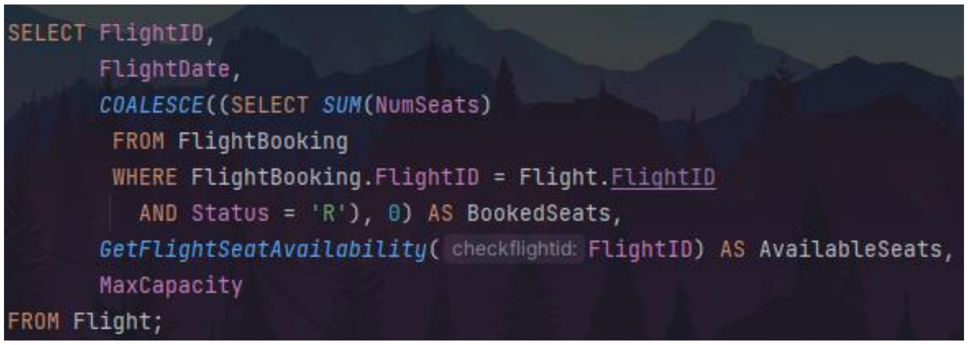
Erroneous testing: Delete customer with reserved booking:



```
rzt22qzu.bookingdb> DELETE
                    FROM LeadCustomer
                    WHERE (CustomerID = 1)
[2023-05-04 18:08:04] [P0001] ERROR: Cannot delete a customer that has a reserved booking.
[2023-05-04 18:08:04] Where: PL/pgSQL function restrictcustomerdeletion() line 7 at RAISE
```

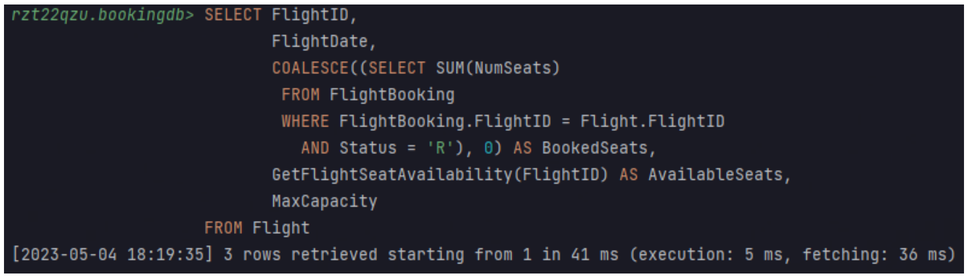
C. Check the availability of seats on all flights by showing the flight ID number, flight date along with the number of booked seats, number of available seats and maximum capacity.

SQL:

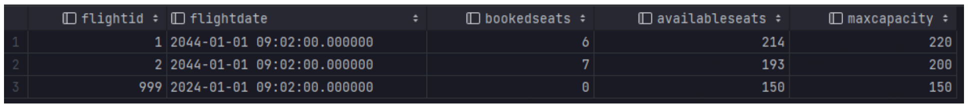


```sql
SELECT FlightID,
       FlightDate,
       COALESCE((SELECT SUM(NumSeats)
        FROM FlightBooking
        WHERE FlightBooking.FlightID = Flight.FlightID
          AND Status = 'R'), 0) AS BookedSeats,
       GetFlightSeatAvailability( checkflightid: FlightID) AS AvailableSeats,
       MaxCapacity
FROM Flight;
```

Output:



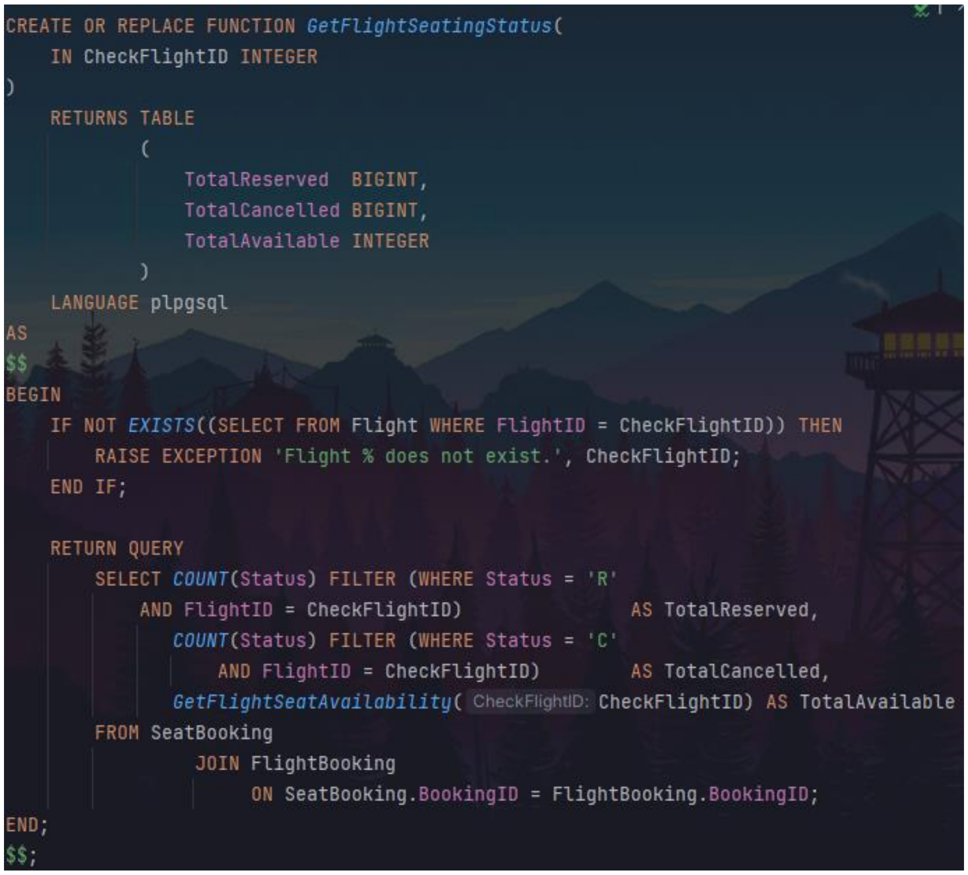
```
rzt22qzu.bookingdb> SELECT FlightID,
                           FlightDate,
                           COALESCE((SELECT SUM(NumSeats)
                            FROM FlightBooking
                            WHERE FlightBooking.FlightID = Flight.FlightID
                              AND Status = 'R'), 0) AS BookedSeats,
                           GetFlightSeatAvailability(FlightID) AS AvailableSeats,
                           MaxCapacity
                    FROM Flight
[2023-05-04 18:19:35] 3 rows retrieved starting from 1 in 41 ms (execution: 5 ms, fetching: 36 ms)
```



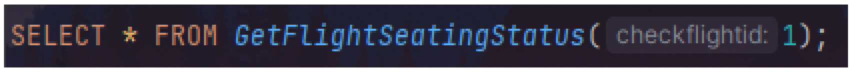
	flightid	flightdate	bookedseats	availableseats	maxcapacity
1	1	2044-01-01 09:02:00.000000	6	214	220
2	2	2044-01-01 09:02:00.000000	7	193	200
3	999	2024-01-01 09:02:00.000000	0	150	150

D. Given a flight ID number, check the status of all seats currently allocated to that flight, i.e., return the total number of reserved/ cancelled/ available seats.

SQL:

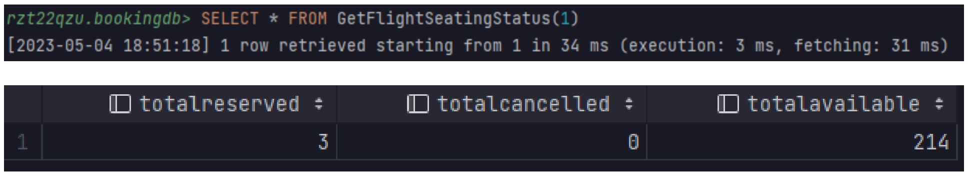


```
CREATE OR REPLACE FUNCTION GetFlightSeatingStatus(
    IN CheckFlightID INTEGER
)
    RETURNS TABLE
            (
                TotalReserved  BIGINT,
                TotalCancelled BIGINT,
                TotalAvailable INTEGER
            )
    LANGUAGE plpgsql
AS
$$
BEGIN
    IF NOT EXISTS((SELECT FROM Flight WHERE FlightID = CheckFlightID)) THEN
        RAISE EXCEPTION 'Flight % does not exist.', CheckFlightID;
    END IF;

    RETURN QUERY
        SELECT COUNT(Status) FILTER (WHERE Status = 'R'
            AND FlightID = CheckFlightID)               AS TotalReserved,
               COUNT(Status) FILTER (WHERE Status = 'C'
                   AND FlightID = CheckFlightID)         AS TotalCancelled,
               GetFlightSeatAvailability( CheckFlightID: CheckFlightID) AS TotalAvailable
        FROM SeatBooking
                 JOIN FlightBooking
                      ON SeatBooking.BookingID = FlightBooking.BookingID;
END;
$$;
```



```
SELECT * FROM GetFlightSeatingStatus( checkflightid: 1);
```

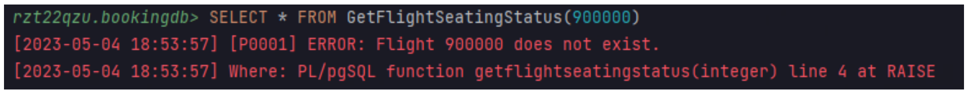
Output:



```
rzt22qzu.bookingdb> SELECT * FROM GetFlightSeatingStatus(1)
[2023-05-04 18:51:18] 1 row retrieved starting from 1 in 34 ms (execution: 3 ms, fetching: 31 ms)
```

	totalreserved	totalcancelled	totalavailable
1	3	0	214

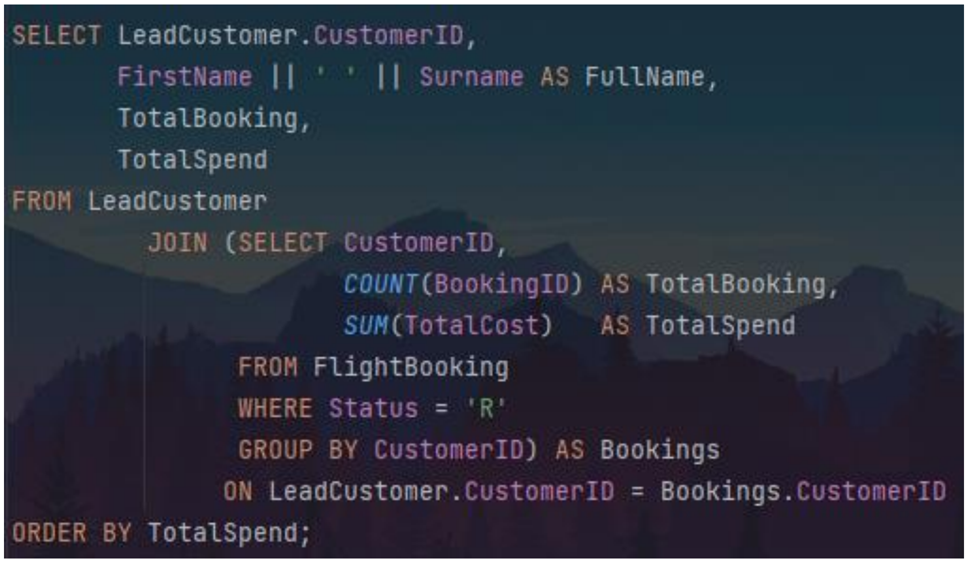
Erroneous testing: Check a flight ID that doesn't exist



```
rzt22qzu.bookingdb> SELECT * FROM GetFlightSeatingStatus(900000)
[2023-05-04 18:53:57] [P0001] ERROR: Flight 900000 does not exist.
[2023-05-04 18:53:57] Where: PL/pgSQL function getflightseatingstatus(integer) line 4 at RAISE
```

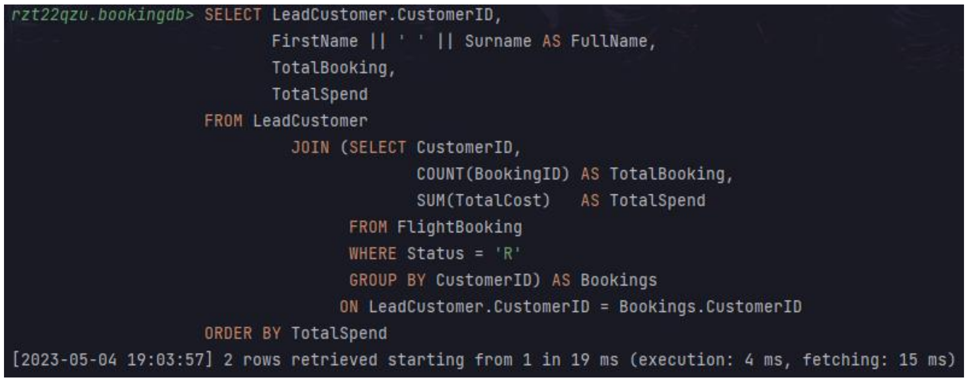
E. Produce a ranked list of all lead customers, showing their ID, their full name, the total number of bookings made, and the total spend made for all bookings. The list should be sorted by decreasing total value.

SQL:

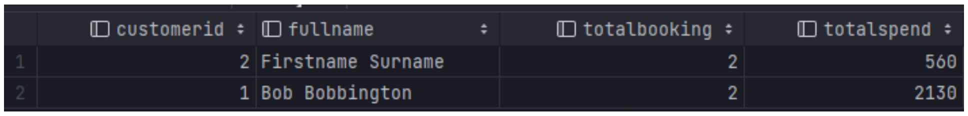


```sql
SELECT LeadCustomer.CustomerID,
       FirstName || ' ' || Surname AS FullName,
       TotalBooking,
       TotalSpend
FROM LeadCustomer
         JOIN (SELECT CustomerID,
                      COUNT(BookingID) AS TotalBooking,
                      SUM(TotalCost)   AS TotalSpend
               FROM FlightBooking
               WHERE Status = 'R'
               GROUP BY CustomerID) AS Bookings
              ON LeadCustomer.CustomerID = Bookings.CustomerID
ORDER BY TotalSpend;
```

Output:



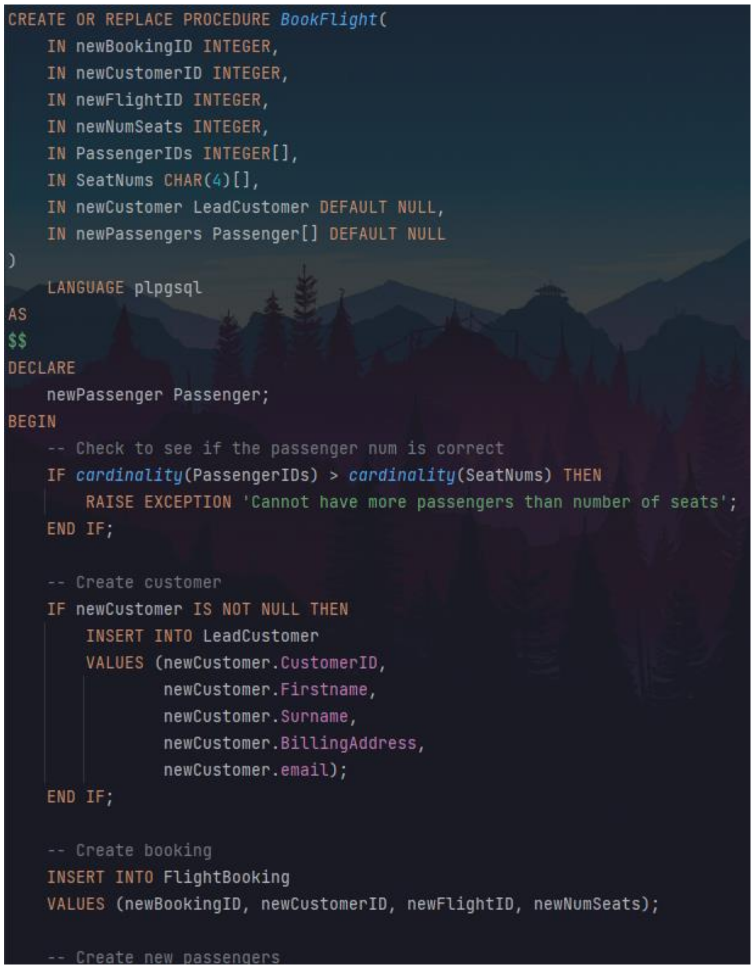
```
rzt22qzu.bookingdb> SELECT LeadCustomer.CustomerID,
                           FirstName || ' ' || Surname AS FullName,
                           TotalBooking,
                           TotalSpend
                    FROM LeadCustomer
                             JOIN (SELECT CustomerID,
                                          COUNT(BookingID) AS TotalBooking,
                                          SUM(TotalCost)   AS TotalSpend
                                   FROM FlightBooking
                                   WHERE Status = 'R'
                                   GROUP BY CustomerID) AS Bookings
                                  ON LeadCustomer.CustomerID = Bookings.CustomerID
                    ORDER BY TotalSpend
[2023-05-04 19:03:57] 2 rows retrieved starting from 1 in 19 ms (execution: 4 ms, fetching: 15 ms)
```



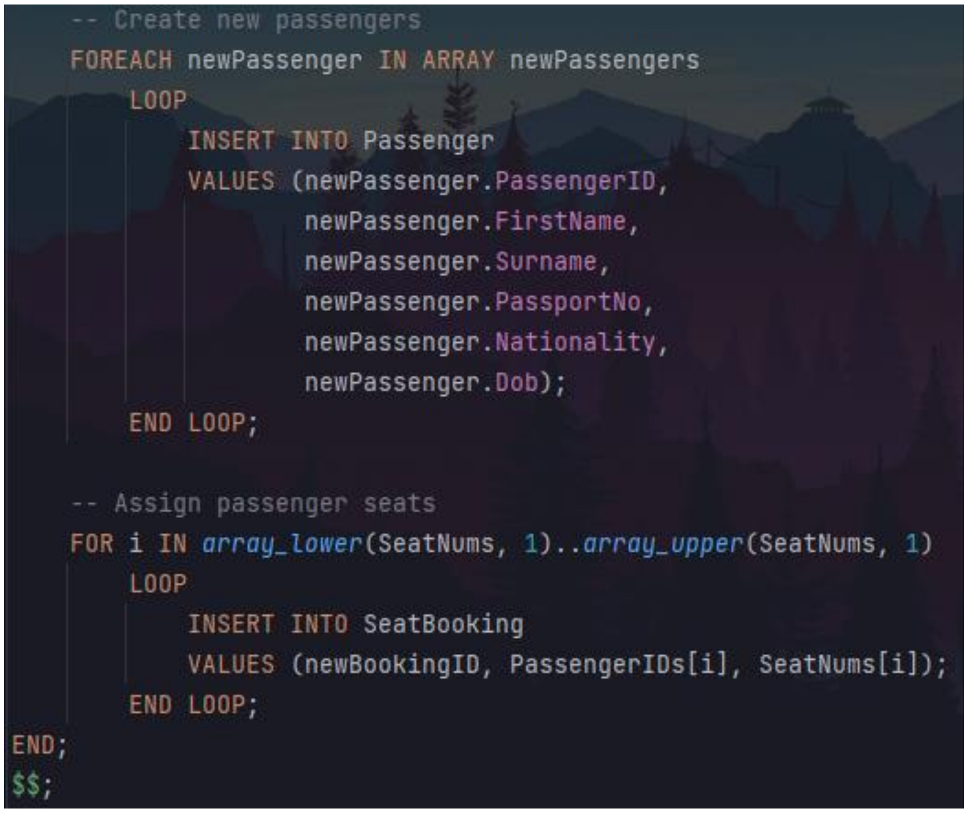
	customerid	fullname	totalbooking	totalspend
1	2	Firstname Surname	2	560
2	1	Bob Bobbington	2	2130

F. Given a booking ID, customer ID number, flight ID number, number of seats required and passenger details, make a booking for a given flight. This procedure should first show seats available in a given flight and then proceed to insert booking, if there are sufficient seats available. The customer could be an existing customer or a new customer, in which case it should be entered first into the database. Seats numbers can be allocated at the time of booking or later on. The making of a booking with all the steps outlined should work as an atomic operation.

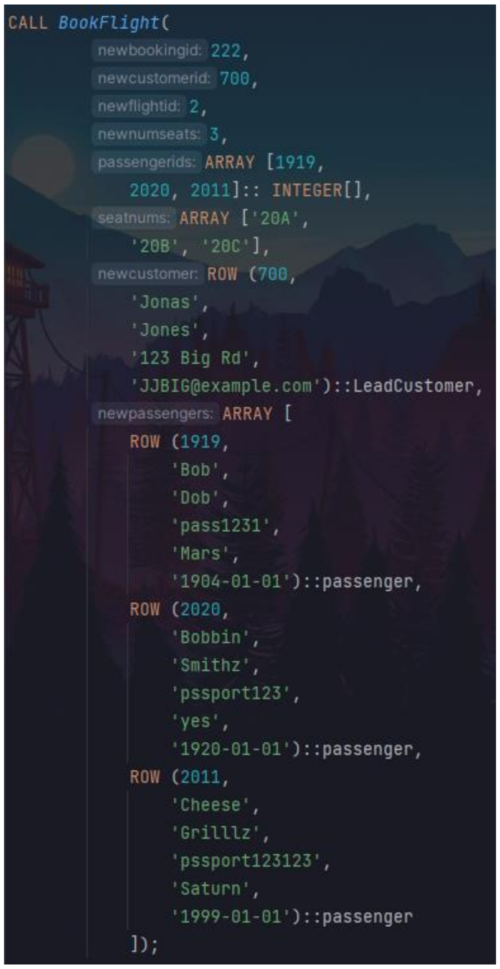
SQL:



```
CREATE OR REPLACE PROCEDURE BookFlight(
    IN newBookingID INTEGER,
    IN newCustomerID INTEGER,
    IN newFlightID INTEGER,
    IN newNumSeats INTEGER,
    IN PassengerIDs INTEGER[],
    IN SeatNums CHAR(4)[],
    IN newCustomer LeadCustomer DEFAULT NULL,
    IN newPassengers Passenger[] DEFAULT NULL
)
    LANGUAGE plpgsql
AS
$$
DECLARE
    newPassenger Passenger;
BEGIN
    -- Check to see if the passenger num is correct
    IF cardinality(PassengerIDs) > cardinality(SeatNums) THEN
        RAISE EXCEPTION 'Cannot have more passengers than number of seats';
    END IF;

    -- Create customer
    IF newCustomer IS NOT NULL THEN
        INSERT INTO LeadCustomer
        VALUES (newCustomer.CustomerID,
                newCustomer.Firstname,
                newCustomer.Surname,
                newCustomer.BillingAddress,
                newCustomer.email);
    END IF;

    -- Create booking
    INSERT INTO FlightBooking
    VALUES (newBookingID, newCustomerID, newFlightID, newNumSeats);

    -- Create new passengers
```

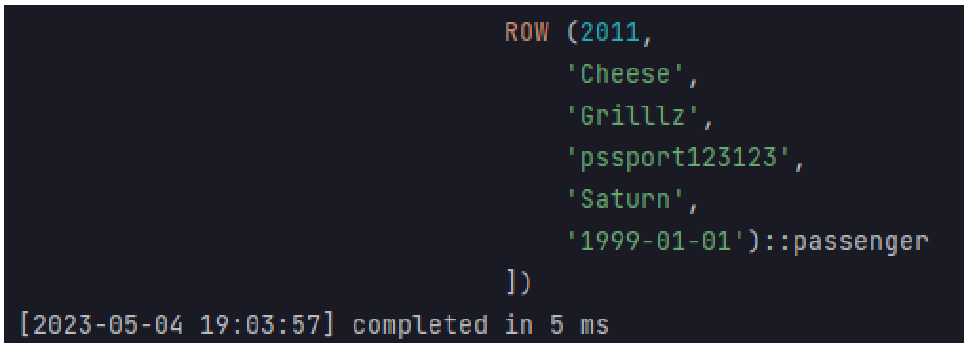


```
    -- Create new passengers
    FOREACH newPassenger IN ARRAY newPassengers
        LOOP
            INSERT INTO Passenger
            VALUES (newPassenger.PassengerID,
                    newPassenger.FirstName,
                    newPassenger.Surname,
                    newPassenger.PassportNo,
                    newPassenger.Nationality,
                    newPassenger.Dob);
        END LOOP;

    -- Assign passenger seats
    FOR i IN array_lower(SeatNums, 1)..array_upper(SeatNums, 1)
        LOOP
            INSERT INTO SeatBooking
            VALUES (newBookingID, PassengerIDs[i], SeatNums[i]);
        END LOOP;
END;
$$;
```

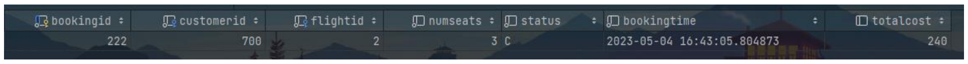


```
CALL BookFlight(
        newbookingid: 222,
        newcustomerid: 700,
        newflightid: 2,
        newnumseats: 3,
        passengerids: ARRAY [1919,
            2020, 2011]:: INTEGER[],
        seatnums: ARRAY ['20A',
            '20B', '20C'],
        newcustomer: ROW (700,
            'Jonas',
            'Jones',
            '123 Big Rd',
            'JJBIG@example.com')::LeadCustomer,
        newpassengers: ARRAY [
            ROW (1919,
                'Bob',
                'Dob',
                'pass1231',
                'Mars',
                '1904-01-01')::passenger,
            ROW (2020,
                'Bobbin',
                'Smithz',
                'pssport123',
                'yes',
                '1920-01-01')::passenger,
            ROW (2011,
                'Cheese',
                'Grilllz',
                'pssport123123',
                'Saturn',
                '1999-01-01')::passenger
            ]);
```

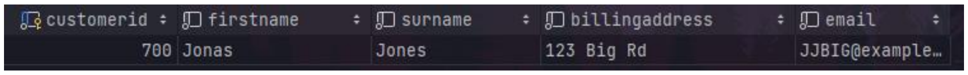
Output:



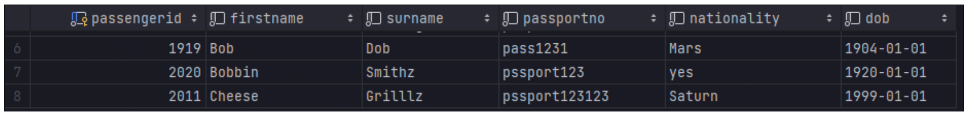
```
                                             ROW (2011,
                                                 'Cheese',
                                                 'Grilllz',
                                                 'pssport123123',
                                                 'Saturn',
                                                 '1999-01-01')::passenger
                                             ])
[2023-05-04 19:03:57] completed in 5 ms
```



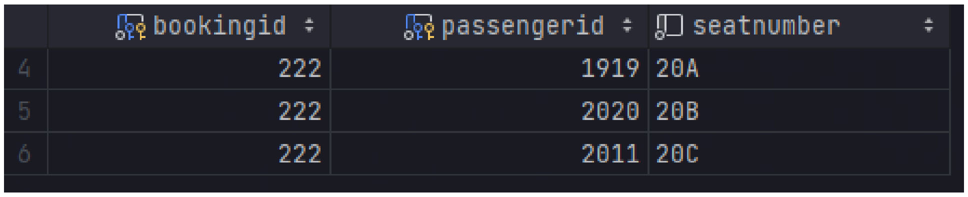
bookingid	customerid	flightid	numseats	status	bookingtime	totalcost
222	700	2	3	C	2023-05-04 16:43:05.804873	240



customerid	firstname	surname	billingaddress	email
700	Jonas	Jones	123 Big Rd	JJBIG@example…

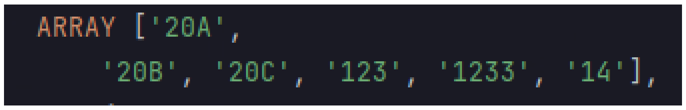


passengerid	firstname	surname	passportno	nationality	dob
1919	Bob	Dob	pass1231	Mars	1904-01-01
2020	Bobbin	Smithz	pssport123	yes	1920-01-01
2011	Cheese	Grilllz	pssport123123	Saturn	1999-01-01

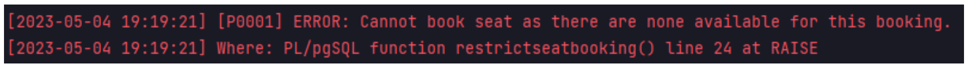


	bookingid	passengerid	seatnumber
4	222	1919	20A
5	222	2020	20B
6	222	2011	20C

Erroneous testing: Provide too many seat numbers:

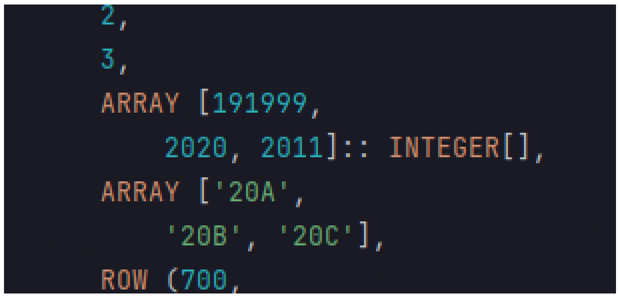


```
ARRAY ['20A',
    '20B', '20C', '123', '1233', '14'],
```

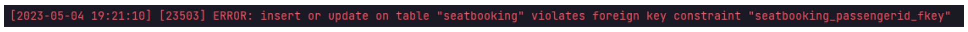


```
[2023-05-04 19:19:21] [P0001] ERROR: Cannot book seat as there are none available for this booking.
[2023-05-04 19:19:21] Where: PL/pgSQL function restrictseatbooking() line 24 at RAISE
```

Erroneous testing: Provide a non-existent passenger ID



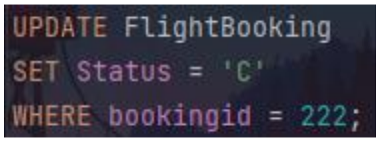
```
    2,
    3,
    ARRAY [191999,
        2020, 2011]:: INTEGER[],
    ARRAY ['20A',
        '20B', '20C'],
    ROW (700,
```



```
[2023-05-04 19:21:10] [23503] ERROR: insert or update on table "seatbooking" violates foreign key constraint "seatbooking_passengerid_fkey"
```

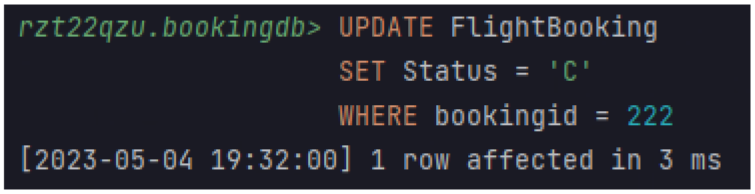
G. Given a booking ID number, cancel the booking. Note that cancelling a booking only changes the status and should not delete the historical details of the original booking. However, cancelled seats should be viewed as available.

SQL:

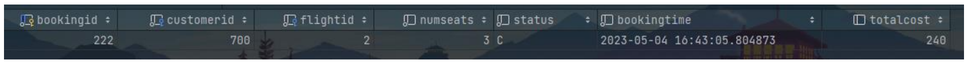


```sql
UPDATE FlightBooking
SET Status = 'C'
WHERE bookingid = 222;
```

Output:



```
rzt22qzu.bookingdb> UPDATE FlightBooking
                    SET Status = 'C'
                    WHERE bookingid = 222
[2023-05-04 19:32:00] 1 row affected in 3 ms
```



bookingid	customerid	flightid	numseats	status	bookingtime	totalcost
222	700	2	3	C	2023-05-04 16:43:05.804873	240