```
SET FOREIGN_KEY_CHECKS=0;

DROP TABLE IF EXISTS weeklyfactor CASCADE;
DROP TABLE IF EXISTS profitfactor CASCADE;
DROP TABLE IF EXISTS ticket CASCADE;
DROP TABLE IF EXISTS booking CASCADE;
DROP TABLE IF EXISTS reserved_pass CASCADE;
DROP TABLE IF EXISTS reservation CASCADE;
DROP TABLE IF EXISTS contact CASCADE;
DROP TABLE IF EXISTS passenger CASCADE;

DROP TABLE IF EXISTS weeklyschedule CASCADE;
DROP TABLE IF EXISTS flight CASCADE;
DROP TABLE IF EXISTS route CASCADE;
DROP TABLE IF EXISTS airport CASCADE;

DROP FUNCTION IF EXISTS calculateFreeSeats;
DROP FUNCTION IF EXISTS calculatePrice;

DROP PROCEDURE IF EXISTS addPayment;
DROP PROCEDURE IF EXISTS addContact;
DROP PROCEDURE IF EXISTS addPassenger;
DROP PROCEDURE IF EXISTS addReservation;
DROP PROCEDURE IF EXISTS addFlight;
DROP PROCEDURE IF EXISTS addRoute;
DROP PROCEDURE IF EXISTS addDestination;
DROP PROCEDURE IF EXISTS addDay;
DROP PROCEDURE IF EXISTS addYear;

DROP TRIGGER IF EXISTS rand_ticketno;
DROP VIEW IF EXISTS allFlights;


SET FOREIGN_KEY_CHECKS=1;


CREATE TABLE airport (
	id VARCHAR(3),
	cityname VARCHAR(30),
	countryname VARCHAR(30),
	CONSTRAINT pk_airport PRIMARY KEY (id)
);

CREATE TABLE route (
	id INT NOT NULL AUTO_INCREMENT,
	routeprice DOUBLE NOT NULL,
      year INT,
      arrival VARCHAR(3),
      departure VARCHAR(3),
	CONSTRAINT pk_route PRIMARY KEY (id)
);



CREATE TABLE weeklyschedule (
	id INT NOT NULL AUTO_INCREMENT,
	year INT NOT NULL,
	day VARCHAR(10),
	time TIME,
    route INT,
        CONSTRAINT pk_weekly PRIMARY KEY (id)
);


CREATE TABLE flight (
        id INT NOT NULL AUTO_INCREMENT,
        week INT NOT NULL,
--      unpaidseats INT NOT NULL,
        scheduleid INT NOT NULL,
        CONSTRAINT pk_flight PRIMARY KEY (id)
);
-- Ändart till bara fullname
CREATE TABLE passenger (
        passportnumber INT NOT NULL,
    fullname VARCHAR(30),
        -- firstname VARCHAR(30),
        -- lastname VARCHAR(30),
        CONSTRAINT pk_passenger PRIMARY KEY (passportnumber)
);

CREATE TABLE contact (
        contact_passportnumber INT NOT NULL,
        email VARCHAR(30),
        phonenumber BIGINT,
    reservationnumber INT,
        CONSTRAINT pk_contact PRIMARY KEY (contact_passportnumber)
);

CREATE TABLE reservation (
        reservationnumber INT NOT NULL AUTO_INCREMENT,
    nopassengers INT NOT NULL,
    resflightno INT NOT NULL,
        CONSTRAINT pk_reservation PRIMARY KEY (reservationnumber)
);

CREATE TABLE reserved_pass (
        rp_resno INT NOT NULL,
    rp_passno INT NOT NULL,
        CONSTRAINT pk_reservedpass PRIMARY KEY (rp_resno, rp_passno)
);

CREATE TABLE booking (
        booking_resno INT NOT NULL,
    cardno BIGINT NOT NULL,
    name_cardholder VARCHAR(30),
    price DOUBLE NOT NULL,

        CONSTRAINT pk_bookingresno PRIMARY KEY (booking_resno)
);

CREATE TABLE ticket (
        ticket_resno INT NOT NULL,
    ticket_passportno INT NOT NULL,
    ticketno INT NOT NULL AUTO_INCREMENT,
    CONSTRAINT pk_ticket PRIMARY KEY (ticketno)
);

CREATE TABLE profitfactor (
        year INT NOT NULL,
    factor DOUBLE NOT NULL,
        CONSTRAINT pk_profitfactor PRIMARY KEY (year)
);

CREATE TABLE weeklyfactor (
        day VARCHAR(10),
    weekly_year INT NOT NULL,
    weekly_factor DOUBLE NOT NULL,
        CONSTRAINT pk_weeklyfactor PRIMARY KEY (day)
);

-- FOREIGN KEYS

ALTER TABLE route ADD CONSTRAINT fk_arrival FOREIGN KEY (arrival)
REFERENCES airport(id);
ALTER TABLE route ADD CONSTRAINT fk_departure FOREIGN KEY (departure)
REFERENCES airport(id);

ALTER TABLE weeklyschedule ADD CONSTRAINT fk_route FOREIGN KEY (route)
REFERENCES route(id);

ALTER TABLE flight ADD CONSTRAINT fk_flightweek FOREIGN KEY (scheduleid)
REFERENCES weeklyschedule (id);

ALTER TABLE contact ADD     CONSTRAINT fk_passenger FOREIGN KEY
(contact_passportnumber) REFERENCES passenger (passportnumber);

ALTER TABLE reservation ADD  CONSTRAINT fk_flightno FOREIGN KEY
(resflightno) REFERENCES flight (id);

ALTER TABLE reserved_pass ADD CONSTRAINT fk_rpresno FOREIGN KEY
(rp_resno) REFERENCES reservation (reservationnumber);
```

```
ALTER TABLE reserved_pass ADD CONSTRAINT fk_rppassno FOREIGN KEY
(rp_passno) REFERENCES passenger (passportnumber);


ALTER TABLE booking ADD CONSTRAINT fk_reservation FOREIGN KEY
(booking_resno) REFERENCES reservation (reservationnumber);
-- ALTER TABLE booking  ADD CONSTRAINT fk_passno FOREIGN KEY
(bcpassportno) REFERENCES contact (contact_passportnumber);


ALTER TABLE ticket ADD CONSTRAINT fk_booking FOREIGN KEY (ticket_resno)
REFERENCES booking (booking_resno);
ALTER TABLE ticket ADD CONSTRAINT fk_passport FOREIGN KEY
(ticket_passportno) REFERENCES passenger (passportnumber);


ALTER TABLE weeklyfactor ADD CONSTRAINT fk_year FOREIGN KEY
(weekly_year) REFERENCES profitfactor (year);




DELIMITER //


CREATE PROCEDURE addYear(IN new_year INT, IN new_factor DOUBLE)
        BEGIN
     INSERT INTO profitfactor(year, factor)
     VALUES (new_year, new_factor);
END;
//


CREATE PROCEDURE addDay(IN new_year INT, IN new_day VARCHAR(10), IN
new_factor DOUBLE)
        BEGIN
     INSERT INTO weeklyfactor(day, weekly_year, weekly_factor)
     VALUES (new_day, new_year, new_factor);
END;
```

```
//

CREATE PROCEDURE addDestination(IN new_id VARCHAR(3), IN new_cityname
VARCHAR(30), IN new_countryname VARCHAR(30))
        BEGIN
    INSERT INTO airport(id, cityname, countryname)
    VALUES (new_id, new_cityname, new_countryname);
END;
//

CREATE PROCEDURE addRoute(IN new_departure_airport VARCHAR(3), IN
new_arrival_airport VARCHAR(3), IN new_year INT, IN new_routeprice DOUBLE)
        BEGIN
        INSERT INTO route(routeprice, year, arrival, departure)
        VALUES (new_routeprice, new_year, new_arrival_airport,
new_departure_airport);
END;
//

CREATE PROCEDURE addFlight(IN new_departure_airport VARCHAR(3), IN
new_arrival_airport VARCHAR(3), IN new_year INT, IN new_day VARCHAR(10), IN
new_departure_time TIME)
        BEGIN
    DECLARE route_id INT;
    DECLARE schedule_id INT;
    DECLARE new_week INT;

    SET route_id = (SELECT id FROM route WHERE arrival=new_arrival_airport
AND departure=new_departure_airport AND year=new_year);
    INSERT INTO weeklyschedule(year, day, time, route)
        VALUES (new_year, new_day, new_departure_time, route_id);
```

```
      SELECT id INTO schedule_id FROM weeklyschedule WHERE
time=new_departure_time AND day=new_day AND year=new_year;

      SET new_week = 1;
      WHILE new_week <= 52 DO
                INSERT INTO flight(week, scheduleid)
                VALUES (new_week, schedule_id);
                SET new_week = new_week + 1;
         END WHILE;
END;
//

CREATE FUNCTION calculateFreeSeats(flight_number INT)
         RETURNS INT
      BEGIN
      DECLARE tickets INT;
      DECLARE free_seats INT;

      SELECT COUNT(*) INTO tickets FROM reservation, ticket WHERE
reservation.reservationnumber = ticket.ticket_resno AND reservation.resflightno =
flight_number;
      SET free_seats = 40 - tickets;
      RETURN free_seats;
END;
//

CREATE FUNCTION calculatePrice(flight_number INT)
         RETURNS    DOUBLE
      BEGIN
      DECLARE seat_price DOUBLE;
      DECLARE booked_seats INT;
```

```
        DECLARE s_id INT;
            DECLARE r_id INT;
        DECLARE route_price DOUBLE;
        DECLARE route_day VARCHAR(10);
        DECLARE route_year INT;
        DECLARE w_factor DOUBLE;
        DECLARE p_factor DOUBLE;

        SET booked_seats = 40 - calculateFreeSeats(flight_number);

        SELECT scheduleid INTO s_id FROM flight WHERE flight.id = flight_number;
        SELECT route INTO r_id FROM weeklyschedule WHERE weeklyschedule.id =
s_id;
        SELECT routeprice INTO route_price FROM route WHERE route.id = r_id;
        SELECT day INTO route_day FROM weeklyschedule WHERE
weeklyschedule.id = s_id;
            SELECT year INTO route_year FROM weeklyschedule WHERE
weeklyschedule.id = s_id;
        SELECT weekly_factor INTO w_factor FROM weeklyfactor WHERE
weeklyfactor.day = route_day AND weeklyfactor.weekly_year = route_year;
        SELECT factor INTO p_factor FROM profitfactor WHERE profitfactor.year =
route_year;

        SET seat_price = route_price * w_factor * (booked_seats + 1) / 40 * p_factor;
        SET seat_price = ROUND(seat_price, 12);

            RETURN seat_price;
END;
//

 CREATE TRIGGER rand_ticketno BEFORE INSERT ON ticket
            FOR EACH ROW
        BEGIN


        SET NEW.ticketno = FLOOR(0 + RAND() * 10000);
END;
//


CREATE PROCEDURE addReservation(IN departure_airport_code VARCHAR(3),
IN arrival_airport_code VARCHAR(3), IN res_year INT, IN res_week INT, IN res_day
VARCHAR(10), IN res_time TIME, IN number_of_passengers INT, OUT
output_reservation_nr INT)
        BEGIN
    DECLARE res_route INT DEFAULT 0;
    DECLARE weekly_schedule_id INT DEFAULT 0;
    DECLARE flight_nr INT DEFAULT 0;
    DECLARE free_seats INT DEFAULT 0;
    DECLARE newresno INT DEFAULT 0;



    SELECT id FROM route WHERE route.arrival=arrival_airport_code AND
route.departure=departure_airport_code AND route.year=res_year INTO res_route;
        SELECT id FROM weeklyschedule WHERE weeklyschedule.route =
res_route AND weeklyschedule.day = res_day AND weeklyschedule.year = res_year
AND weeklyschedule.time = res_time INTO weekly_schedule_id;
    SELECT id FROM flight WHERE flight.scheduleid = weekly_schedule_id AND
flight.week = res_week INTO flight_nr;


    SET free_seats = calculateFreeSeats(flight_nr);




    IF res_route != 0 AND weekly_schedule_id != 0 AND flight_nr != 0 THEN
```

```
                 IF free_seats >= number_of_passengers THEN
                 SET newresno = FLOOR(rand() * 23326);

                         INSERT INTO reservation (reservationnumber, nopassengers,
resflightno) VALUES (newresno, number_of_passengers, flight_nr);

               SET output_reservation_nr = newresno;
               SELECT * from reservation;
               SELECT output_reservation_nr as "lasrt insert";
               SELECT 'OK result' AS 'Message';
                 ELSE
          SELECT 'There are not enough seats available on the chosen flight' AS
'Message';
          END IF;
        ELSE
     SELECT 'There exist no flight for the given route, date and time' AS 'Message';
     END IF;
END;
//
CREATE PROCEDURE addPassenger(IN reservation_nr INT, IN passport_number
INT, IN passenger_name VARCHAR(30))
BEGIN

IF EXISTS( SELECT reservationnumber FROM reservation WHERE
reservation.reservationnumber = reservation_nr) THEN


        IF NOT EXISTS ( SELECT booking_resno FROM booking WHERE
booking.booking_resno = reservation_nr) THEN
```

```
                    IF NOT EXISTS ( SELECT passportnumber FROM passenger WHERE
passenger.passportnumber = passport_number AND passenger.fullname =
passenger_name) THEN



                    INSERT INTO passenger (passportnumber, fullname) VALUES
(passport_number, passenger_name);

                    INSERT INTO reserved_pass (rp_resno, rp_passno) values
(reservation_nr, passport_number);


          -- SELECT "OK result" AS "Message";

          ELSE

                    INSERT INTO reserved_pass (rp_resno, rp_passno) values
(reservation_nr, passport_number);

                    -- SELECT "OK result" AS "Message";

      END IF;


          ELSE

      SELECT "The booking has already been payed and no futher passengers can
be added" AS "Message";

      END IF;


ELSE

          SELECT 'The given reservation number does not exist' AS 'Message';

END IF;

END;

//


CREATE PROCEDURE addContact( IN reservation_nr INT, IN passport_number
INT, IN email VARCHAR(30), IN phone INT)

BEGIN
```

```
IF EXISTS( SELECT reservationnumber, passportnumber FROM reservation,
passenger WHERE reservation.reservationnumber = reservation_nr) THEN


    IF EXISTS (SELECT rp_resno FROM reserved_pass WHERE rp_resno =
reservation_nr AND rp_passno = passport_number) THEN
            INSERT INTO contact (contact_passportnumber, email, phonenumber,
reservationnumber) VALUES (passport_number, email, phone, reservation_nr);
            SELECT "OK result addContact" AS "Message";
    ELSE
            SELECT "The person is not a passenger of the reservation" AS
"Message";
        END IF;


ELSE
        SELECT "The given reservation number does not exist" AS "Message";



END IF;
END;
//



CREATE PROCEDURE addPayment (IN reservation_nr INT, IN cardholder_name
VARCHAR(30), IN credit_card_number BIGINT)
BEGIN


DECLARE flightno INT DEFAULT 0;
DECLARE price DOUBLE DEFAULT 0;
DECLARE no_of_pass INT DEFAULT 0;
DECLARE ticket_number INT;
```

```
        IF EXISTS( SELECT reservationnumber from reservation where
reservation.reservationnumber = reservation_nr ) THEN
                IF EXISTS ( SELECT contact_passportnumber FROM contact WHERE
contact.reservationnumber = reservation_nr) THEN
                        SELECT resflightno FROM reservation WHERE
reservation.reservationnumber = reservation_nr INTO flightno;
                        SELECT count(*) FROM reserved_pass WHERE
reserved_pass.rp_resno = reservation_nr INTO no_of_pass;




              IF no_of_pass <= calculateFreeSeats(flightno) THEN
                                SELECT SLEEP(15);
                                SET price = no_of_pass*calculatePrice(flightno);
                                        INSERT INTO booking (booking_resno, cardno,
name_cardholder, price) VALUES (reservation_nr, credit_card_number,
cardholder_name, price);
                        INSERT INTO ticket(ticket_resno, ticket_passportno) SELECT *
FROM reserved_pass WHERE reserved_pass.rp_resno = reservation_nr;


                                SELECT "OK Result addPayment" AS "Message";
              ELSE
                                SELECT "There are not enough seats available on the
flight anymore, deleting reservation" AS "Message";
                                DELETE FROM reserved_pass WHERE
reserved_pass.rp_resno = reservation_nr;
                   SELECT * from booking;
                   SELECT * from reservation;
                   SELECT "efter respass" AS "kuken";
                                DELETE FROM reservation WHERE
reservation.reservationnumber = reservation_nr;
                   SELECT "efter respass och reservation" AS "blablavla";
              END IF;
                ELSE
```

```
                        SELECT "The reservation has no contact yet" AS "Message";


                END IF;
        ELSE
                SELECT "The given reservation number does not exist" AS "Message";
        END IF;


 END;
//


CREATE VIEW allFlights AS
        SELECT depart.cityname as departure_city_name, arrive.cityname as
destination_city_name, ws.time as departure_time, ws.day as departure_day,
fl.week as departure_week, ws.year as departure_year, calculateFreeSeats(fl.id) AS
nr_of_free_seats, calculatePrice(fl.id) AS current_price_per_seat
        FROM flight AS fl INNER JOIN weeklyschedule AS ws INNER JOIN route as
ro INNER JOIN airport as depart INNER JOIN airport as arrive
     WHERE fl.scheduleid = ws.id AND ws.route = ro.id AND ro.arrival = arrive.id
AND ro.departure = depart.id;
//
SELECT * from route;
SELECT * from allFlights;
```