```
################################################################
#Drop all tables, functions, unlock all tables if crash etc.
################################################################
SET FOREIGN_KEY_CHECKS = 0;
UNLOCK TABLES;
DROP TABLE IF EXISTS PASSENGER;
DROP TABLE IF EXISTS CONTACT;
DROP TABLE IF EXISTS RESERVATION;
DROP TABLE IF EXISTS RESERVED;
DROP TABLE IF EXISTS FLIGHT;
DROP TABLE IF EXISTS WEEKLYSCHEDULE;
DROP TABLE IF EXISTS WEEKDAY;
DROP TABLE IF EXISTS BOOKING;
DROP TABLE IF EXISTS YEAR;
DROP TABLE IF EXISTS CREDENTIALS;
DROP TABLE IF EXISTS ROUTE;
DROP TABLE IF EXISTS AIRPORT;

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

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

DROP VIEW IF EXISTS allFlights;
SET FOREIGN_KEY_CHECKS = 1;

##################
#Create all tables
##################

CREATE TABLE PASSENGER(
      PASSPORTNR INTEGER NOT NULL,
    NAME VARCHAR(30),
    CONSTRAINT PK_PASSENGER PRIMARY KEY(PASSPORTNR)
    );

CREATE TABLE CONTACT(
      PASSPORTNR INTEGER NOT NULL,
      EMAIL VARCHAR(30),
    PHONENUMBER BIGINT,
    CONSTRAINT PK_CONTACT PRIMARY KEY(PASSPORTNR)
    );

CREATE TABLE RESERVATION(
      RESERVATIONID INTEGER NOT NULL AUTO_INCREMENT,
    FLIGHT INTEGER,
    CONTACT INTEGER,
    PASSENGERS INTEGER,
    CONSTRAINT PK_RESERVATION PRIMARY KEY(RESERVATIONID)
    );
```

```
CREATE TABLE RESERVED(
      PASSPORTNR INTEGER NOT NULL,
    RESERVATIONID INTEGER NOT NULL,
    TICKETNR INTEGER,
    PRIMARY KEY(PASSPORTNR, RESERVATIONID)
      );

CREATE TABLE FLIGHT (
      FLIGHTNUMBER INTEGER NOT NULL AUTO_INCREMENT,
    WEEKLYFLIGHT INTEGER NOT NULL,
    WEEK INTEGER,
    CONSTRAINT PK_FLIGHT PRIMARY KEY(FLIGHTNUMBER)
    );

CREATE TABLE WEEKLYSCHEDULE(
      FLIGHTID INTEGER NOT NULL AUTO_INCREMENT,
    WEEKDAY VARCHAR(10) NOT NULL,
    YEAR INTEGER NOT NULL,
    DEPARTURETIME TIME,
    ROUTE INTEGER NOT NULL,
    CONSTRAINT PK_WEEKLYSCHEDULE PRIMARY KEY(FLIGHTID)
    );

CREATE TABLE WEEKDAY(
      WEEKDAY VARCHAR(10) NOT NULL,
    DAYPRICEFACTOR DOUBLE NOT NULL,
    YEAR INTEGER NOT NULL,
    CONSTRAINT PK_WEEKDAY PRIMARY KEY(WEEKDAY,YEAR)
    );

CREATE TABLE BOOKING(
      RESERVATIONID INTEGER NOT NULL,
    TOTALPRICE DOUBLE,
    PAYINGCARD BIGINT NOT NULL,
    CONSTRAINT PK_BOOKING PRIMARY KEY(RESERVATIONID)
    );

CREATE TABLE YEAR(
      YEAR INTEGER NOT NULL,
    PRICEFACTOR DOUBLE,
    CONSTRAINT PK_YEAR PRIMARY KEY(YEAR));

CREATE TABLE CREDENTIALS(
      CARDNUMBER BIGINT NOT NULL,
    NAME VARCHAR(30),
    CONSTRAINT PK_CREDENTIALS PRIMARY KEY(CARDNUMBER));

CREATE TABLE ROUTE(
      ROUTEID INTEGER NOT NULL AUTO_INCREMENT,
    DEPARTURE VARCHAR(3) NOT NULL,
    ARRIVAL VARCHAR(3) NOT NULL,
    ROUTEPRICE DOUBLE,
    YEAR INTEGER NOT NULL,
    CONSTRAINT PK_ROUTE PRIMARY KEY(ROUTEID)
    );

CREATE TABLE AIRPORT(
      AIRPORTCODE VARCHAR(3) NOT NULL,
    AIRPORTNAME VARCHAR(30) NOT NULL,
```

```
    COUNTRY VARCHAR(30),
    CONSTRAINT PK_AIRPORT PRIMARY KEY(AIRPORTCODE));

##############
#FOREIGN KEYS
##############
ALTER TABLE ROUTE ADD CONSTRAINT FK_ROUTE_DEPARTURE FOREIGN KEY(DEPARTURE)
REFERENCES AIRPORT(AIRPORTCODE);
ALTER TABLE ROUTE ADD CONSTRAINT FK_ROUTE_ARRIVAL FOREIGN KEY(ARRIVAL) REFERENCES
AIRPORT(AIRPORTCODE);
ALTER TABLE ROUTE ADD CONSTRAINT FK_ROUTE_YEAR FOREIGN KEY(YEAR) REFERENCES
YEAR(YEAR);

ALTER TABLE CONTACT ADD CONSTRAINT FK_CONTACT FOREIGN KEY(PASSPORTNR) REFERENCES
PASSENGER(PASSPORTNR);

ALTER TABLE RESERVATION ADD CONSTRAINT FK_RESERVATION_FLIGHT FOREIGN KEY(FLIGHT)
REFERENCES FLIGHT(FLIGHTNUMBER);
ALTER TABLE RESERVATION ADD CONSTRAINT FK_RESERVATION_CONTACT FOREIGN KEY(CONTACT)
REFERENCES CONTACT(PASSPORTNR);

ALTER TABLE RESERVED ADD CONSTRAINT FK_RESERVED_PASSPORT FOREIGN KEY (PASSPORTNR)
REFERENCES PASSENGER(PASSPORTNR);
ALTER TABLE RESERVED ADD CONSTRAINT FK_RESERVED_RESERVATION FOREIGN KEY
(RESERVATIONID) REFERENCES RESERVATION(RESERVATIONID);

ALTER TABLE FLIGHT ADD CONSTRAINT FK_FLIGHT FOREIGN KEY(WEEKLYFLIGHT) REFERENCES
WEEKLYSCHEDULE(FLIGHTID);

ALTER TABLE WEEKLYSCHEDULE ADD CONSTRAINT FK_WEEKLYSCHEDULE_WEEKDAY FOREIGN
KEY(WEEKDAY) REFERENCES WEEKDAY(WEEKDAY);
ALTER TABLE WEEKLYSCHEDULE ADD CONSTRAINT FK_WEEKLYSCHEDULE_YEAR FOREIGN KEY(YEAR)
REFERENCES YEAR(YEAR);
ALTER TABLE WEEKLYSCHEDULE ADD CONSTRAINT FK_WEEKLYSCHEDULE_ROUTE FOREIGN
KEY(ROUTE) REFERENCES ROUTE(ROUTEID);

ALTER TABLE WEEKDAY ADD CONSTRAINT FK_WEEKDAY FOREIGN KEY(YEAR) REFERENCES
YEAR(YEAR);

ALTER TABLE BOOKING ADD CONSTRAINT FK_BOOKING_RESERVATION FOREIGN
KEY(RESERVATIONID) REFERENCES RESERVATION(RESERVATIONID);
ALTER TABLE BOOKING ADD CONSTRAINT FK_BOOKING_PAYINGCARD FOREIGN KEY(PAYINGCARD)
REFERENCES CREDENTIALS(CARDNUMBER);

#############
# Procedures
#############
DELIMITER //

CREATE PROCEDURE addYear(year INTEGER, pricefactor DOUBLE)
                     BEGIN
                                  INSERT INTO YEAR VALUES(year,pricefactor);

                     END //

 CREATE PROCEDURE addDay(year INTEGER, weekday VARCHAR(10), pricefactor DOUBLE)
                     BEGIN
                                  INSERT INTO WEEKDAY
VALUES(weekday,pricefactor,year);
```

```
                    END //

CREATE PROCEDURE addDestination(airportcode VARCHAR(3), name_id VARCHAR(30),
country_id VARCHAR(30))
                    BEGIN
                                INSERT INTO AIRPORT
VALUES(airportcode,name_id,country_id);
                    END //

CREATE PROCEDURE addRoute(departureAirportCode VARCHAR(3), arrivalAirportCode
VARCHAR(3), year INTEGER, routePrice DOUBLE)
                    BEGIN
                                INSERT INTO ROUTE(DEPARTURE, ARRIVAL,
ROUTEPRICE, YEAR) VALUES(departureAirportCode, arrivalAirportCode, routePrice,
year);
                    END //

CREATE PROCEDURE addFlight(IN departureAirportCode VARCHAR(3),IN arrivalAirportCode
VARCHAR(3),IN year INTEGER,IN weekday VARCHAR(10),IN departureTime TIME)
BEGIN
      DECLARE cnt INT DEFAULT 1;
      DECLARE found_id INTEGER;
      SELECT ROUTEID INTO found_id FROM ROUTE AS R WHERE R.DEPARTURE =
departureAirportCode AND R.ARRIVAL = arrivalAirportCode AND R.YEAR = year;
      INSERT INTO WEEKLYSCHEDULE (WEEKDAY, YEAR, DEPARTURETIME, ROUTE) VALUES
(weekday, year, departureTime, found_id);
            WHILE cnt < 53 DO
                  INSERT INTO FLIGHT(WEEK, WEEKLYFLIGHT) VALUES
                        (cnt, (SELECT W.FLIGHTID FROM WEEKLYSCHEDULE AS W WHERE
W.YEAR = year and W.DEPARTURETIME = departureTime AND W.ROUTE IN
                                          (SELECT ROUTEID FROM ROUTE AS R WHERE
R.ARRIVAL = arrivalAirportCode AND R.DEPARTURE = departureAirportCode and R.YEAR =
year)
                                                      AND W.WEEKDAY IN (SELECT
WEEKDAY FROM WEEKDAY WD WHERE WD.WEEKDAY = weekday AND WD.YEAR = year)));

                  SET cnt = cnt + 1;
            END WHILE;
END; //

CREATE PROCEDURE addReservation(IN departure_airportcode VARCHAR(3),IN
arrival_airportcode VARCHAR(3), IN year INTEGER, week INTEGER, IN day VARCHAR(10),

                                                      time TIME, IN
number_passengers INTEGER , OUT output_reservation_nr INTEGER)
BEGIN
      DECLARE flight_number INT DEFAULT NULL;
    DECLARE res_number INT;

    SELECT FLIGHTNUMBER INTO flight_number FROM FLIGHT F WHERE F.WEEK = week AND
F.WEEKLYFLIGHT IN
                                (SELECT FLIGHTID FROM WEEKLYSCHEDULE W WHERE
W.DEPARTURETIME = time AND W.YEAR = year AND W.WEEKDAY = day AND ROUTE IN(
                                                      (SELECT ROUTEID FROM ROUTE R
WHERE R.YEAR = year and R.DEPARTURE = departure_airportcode AND R.ARRIVAL =
arrival_airportcode)));

    #Does the flight exists?
      IF flight_number IS NULL
```

```
      THEN
            SELECT "There exist no flight for the given route, date and time" as
"Message";
      ELSE
            IF number_passengers > calculateFreeSeats(flight_number)
            THEN
                  SELECT "There are not enough seats available on the chosen
flight" as "Message";
            ELSE
                  SET res_number = rand() * 10000;
                  INSERT INTO RESERVATION(RESERVATIONID, FLIGHT, PASSENGERS) VALUES
(res_number, flight_number, number_passengers);
                  SET output_reservation_nr = res_number;
            END IF;
      END IF;
END;
//
CREATE PROCEDURE addPassenger(IN reservation_nr INTEGER, IN passport_number
INTEGER, IN name VARCHAR(30))

BEGIN
      DECLARE not_valid INT default NULL;
    DECLARE already_passenger INT DEFAULT NULL;
    DECLARE already_booked INT DEFAULT NULL;

    SELECT RESERVATIONID INTO not_valid  FROM RESERVATION WHERE RESERVATIONID =
reservation_nr;
    SELECT PASSPORTNR INTO already_passenger FROM PASSENGER WHERE PASSPORTNR =
passport_number;
    SELECT TICKETNR INTO already_booked FROM RESERVED WHERE RESERVATIONID =
reservation_nr LIMIT 1;

    IF not_valid IS NULL
    THEN
            SELECT "The given reservation number does not exist" AS 'MESSAGE';
    ELSE
            IF already_booked IS NULL
        THEN
                  IF already_passenger IS NULL
                  THEN
                        INSERT INTO PASSENGER(PASSPORTNR, NAME) VALUES
(passport_number, name);
                        INSERT INTO RESERVED(PASSPORTNR, RESERVATIONID) VALUES
(passport_number, reservation_nr);
                        SELECT 'Passenger added' AS 'Message';
                  ELSE
                        INSERT INTO RESERVED(PASSPORTNR, RESERVATIONID) VALUES
(passport_number, reservation_nr);
                        SELECT 'Passenger added' AS 'Message';
                  END IF;
            ELSE
                  SELECT "The booking has already been payed and no futher
passengers can be added" AS 'Message';
            END IF;
    END IF;
END;
//
CREATE PROCEDURE addContact(IN reservation_nr INTEGER, IN passport_number INTEGER,
IN email VARCHAR(30), IN phone BIGINT)
```

```
BEGIN
      DECLARE not_valid_res INT DEFAULT NULL;
    DECLARE not_valid_ticket_res INT DEFAULT NULL;
    DECLARE already_contact INT DEFAULT NULL;

    SELECT RESERVATIONID INTO not_valid_res FROM RESERVATION WHERE RESERVATIONID =
reservation_nr;
    SELECT PASSPORTNR INTO not_valid_ticket_res FROM RESERVED WHERE PASSPORTNR =
passport_number AND RESERVATIONID = reservation_nr;
      SELECT PASSPORTNR INTO already_contact FROM CONTACT WHERE PASSPORTNR =
passport_number;

      IF not_valid_res IS NULL
            THEN
                  SELECT "The given reservation number does not exist" AS
'MESSAGE';
            ELSE
                  IF not_valid_ticket_res IS NULL
                        THEN
                              SELECT "The person is not a passenger of the
reservation" AS 'MESSAGE';
                        ELSE
                IF already_contact IS NULL
                              THEN
                                    INSERT INTO
CONTACT(PASSPORTNR,EMAIL,PHONENUMBER) VALUES(passport_number, email, phone);
                                    UPDATE RESERVATION
                                    SET CONTACT = passport_number
                                    WHERE RESERVATIONID = reservation_nr;
                                    SELECT 'Contact added' AS 'MESSAGE';
                              ELSE
                                    UPDATE RESERVATION
                                    SET CONTACT = passport_number
                                    WHERE RESERVATIONID = reservation_nr;
                                    SELECT 'Contact added' AS 'MESSAGE';
                        END IF;
                  END IF;
    END IF;

END;
//

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

BEGIN
      DECLARE reservation_contact INTEGER DEFAULT NULL;
      DECLARE flight_number INT DEFAULT NULL;
    DECLARE price_total DOUBLE DEFAULT NULL;
    DECLARE number_of_passengers INTEGER;
    DECLARE existing_reservation INTEGER DEFAULT NULL;
    DECLARE already_booked INTEGER DEFAULT NULL;
    DECLARE already_credential DOUBLE DEFAULT NULL;

      SELECT CONTACT INTO reservation_contact FROM RESERVATION WHERE RESERVATIONID
= reservation_nr;
    SELECT FLIGHT INTO flight_number FROM RESERVATION WHERE RESERVATIONID =
reservation_nr;
```

```
    SELECT RESERVATIONID INTO existing_reservation FROM RESERVATION WHERE
RESERVATIONID = reservation_nr;
      SET price_total = calculatePrice(flight_number);
    SELECT COUNT(*) INTO number_of_passengers FROM RESERVED WHERE RESERVATIONID =
reservation_nr;
    SELECT RESERVATIONID INTO already_booked FROM BOOKING WHERE RESERVATIONID =
reservation_nr;
    SELECT CARDNUMBER INTO already_credential FROM CREDENTIALS WHERE NAME =
cardholder_name;

   IF existing_reservation IS NULL
            THEN
        SELECT 'The given reservation number does not exist' AS 'Message';
        ELSE
            IF reservation_contact IS NULL
            THEN
                  SELECT "The reservation has no contact yet" AS 'MESSAGE';
            ELSE
                  #Are there enough seats?
                  IF number_of_passengers > calculateFreeSeats(flight_number)
                  THEN
                        SELECT 'There are not enough seats available on the flight
anymore, deleting reservation' AS 'MESSAGE';
                        DELETE FROM RESERVED WHERE RESERVATIONID = reservation_nr;
                DELETE FROM RESERVATION WHERE RESERVATIONID = reservation_nr;
                  ELSE
                        IF already_booked IS NOT NULL
                THEN
                              SELECT 'The reservation has already been paid' AS
'Message';
                ELSE
                              IF already_credential IS NULL
                              THEN
                                    INSERT INTO CREDENTIALS
VALUES(credit_card_number, cardholder_name);
                         SELECT SLEEP(5);
                                    INSERT INTO BOOKING VALUES (reservation_nr,
price_total, credit_card_number);
                                    UPDATE RESERVED
                                    SET TICKETNR = rand() * 100000
                                    WHERE RESERVATIONID = reservation_nr;

                                    SELECT 'Payment complete' AS 'MESSAGE';
                              ELSE
                                    INSERT INTO BOOKING VALUES (reservation_nr,
price_total, credit_card_number);
                                    UPDATE RESERVED
                                    SET TICKETNR = rand() * 100000
                                    WHERE RESERVATIONID = reservation_nr;

                                    SELECT 'Payment complete' AS 'MESSAGE';
                              END IF;
                        END IF;
                  END IF;
        END IF;
    END IF;
END;
//
```

```
################
# FUNCTIONS
################


CREATE FUNCTION calculateFreeSeats(flightnumber integer)
RETURNS INT
BEGIN
      DECLARE payed_seats INTEGER;
      SELECT COUNT(*) INTO payed_seats FROM RESERVED WHERE TICKETNR IS NOT NULL AND
RESERVATIONID IN (SELECT RESERVATIONID FROM RESERVATION WHERE FLIGHT =
flightnumber);
    RETURN 40 - payed_seats;
END;
//

CREATE FUNCTION calculatePrice(flightnumber integer)
RETURNS DOUBLE
BEGIN
    DECLARE route_price DOUBLE;
    DECLARE weekday_factor DOUBLE;
    DECLARE booked_passengers integer;
    DECLARE profit_factor DOUBLE;
    DECLARE flight_year INTEGER;
    DECLARE flight_day VARCHAR(10);

    SELECT YEAR INTO flight_year FROM WEEKLYSCHEDULE WHERE FLIGHTID = flightnumber;
    SELECT WEEKDAY INTO flight_day FROM WEEKLYSCHEDULE WHERE FLIGHTID =
flightnumber;
      SELECT ROUTEPRICE INTO route_price FROM ROUTE WHERE ROUTEID IN
                        (SELECT ROUTE FROM WEEKLYSCHEDULE WHERE FLIGHTID =
flightnumber);
    SELECT DAYPRICEFACTOR INTO weekday_factor FROM WEEKDAY WHERE WEEKDAY =
flight_day AND YEAR = flight_year;

    SET booked_passengers = 40 - calculateFreeSeats(flightnumber);

    SELECT PRICEFACTOR INTO profit_factor FROM YEAR WHERE YEAR = flight_year;

      #Totalprice = routeprice to/from * weekdayfactorday * (bookedPASSENGERsflight
+1)/40 *profitfactor

      RETURN route_price * weekday_factor * ((booked_passengers  + 1) /40) *
profit_factor;
END;
//

############
# TRIGGER
############

CREATE TRIGGER randomTicketNr before UPDATE ON RESERVED
FOR EACH ROW
BEGIN
      SET new.TICKETNR = FLOOR((rand() * 100000 ));
END;
//
delimiter ;
############
```

```
# VIEWS
############

CREATE VIEW allFlights AS
      SELECT route.DEPARTURE AS 'departure_city_name',
      route.ARRIVAL AS 'destination_city_name',
      ws.DEPARTURETIME AS 'departure_time',
      ws.WEEKDAY AS 'departure_day',
      flight.WEEK AS 'departure_week',
      ws.YEAR AS 'departure_year',
      calculateFreeSeats(flight.FLIGHTNUMBER) AS 'nr_of_free_seats',
      calculatePrice(flight.FLIGHTNUMBER) AS 'current_price_per_seat'
    FROM ROUTE route, WEEKLYSCHEDULE ws, FLIGHT flight
    WHERE route.ROUTEID = ws.ROUTE AND flight.FLIGHTNUMBER = ws.FLIGHTID;
```