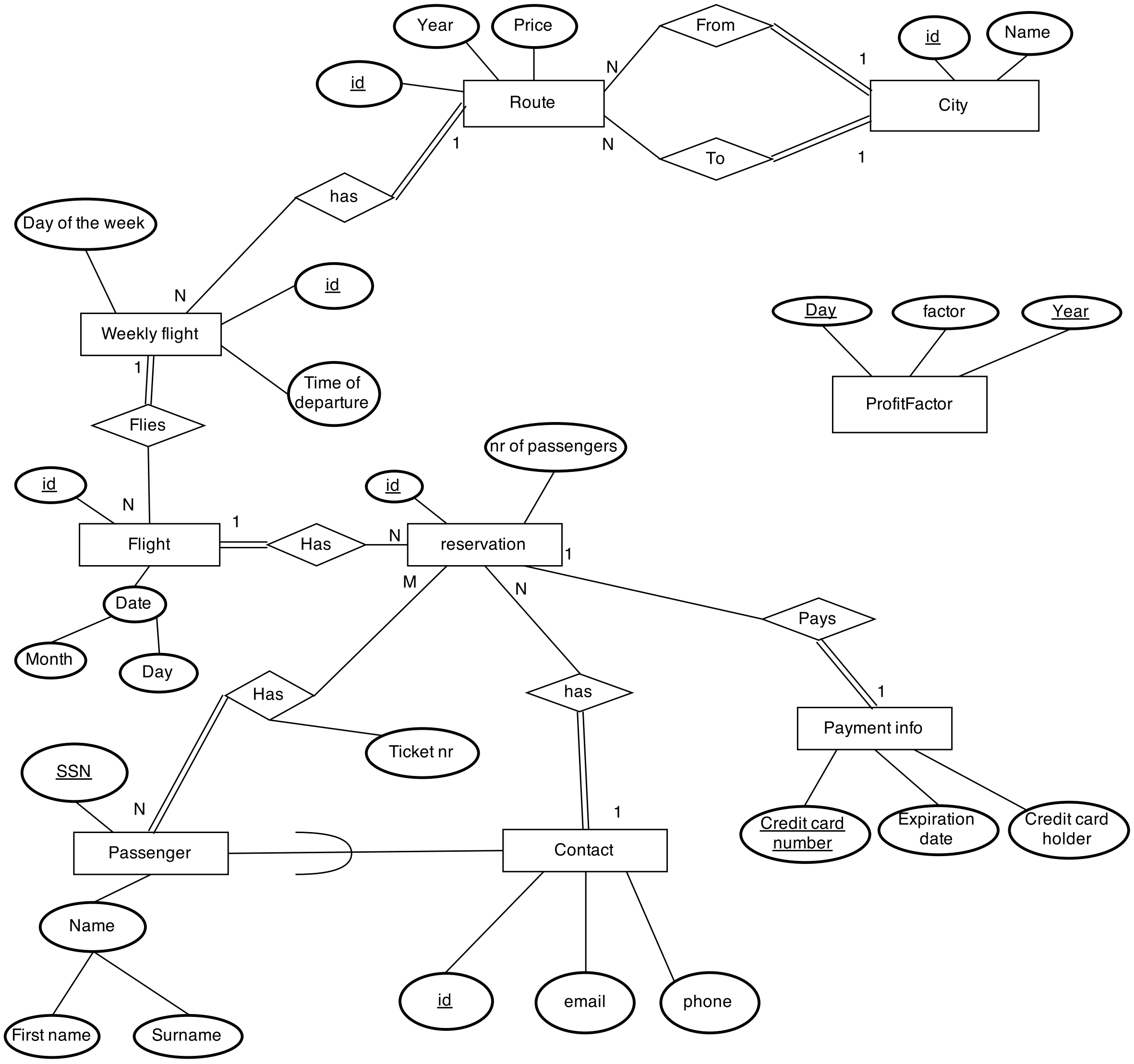
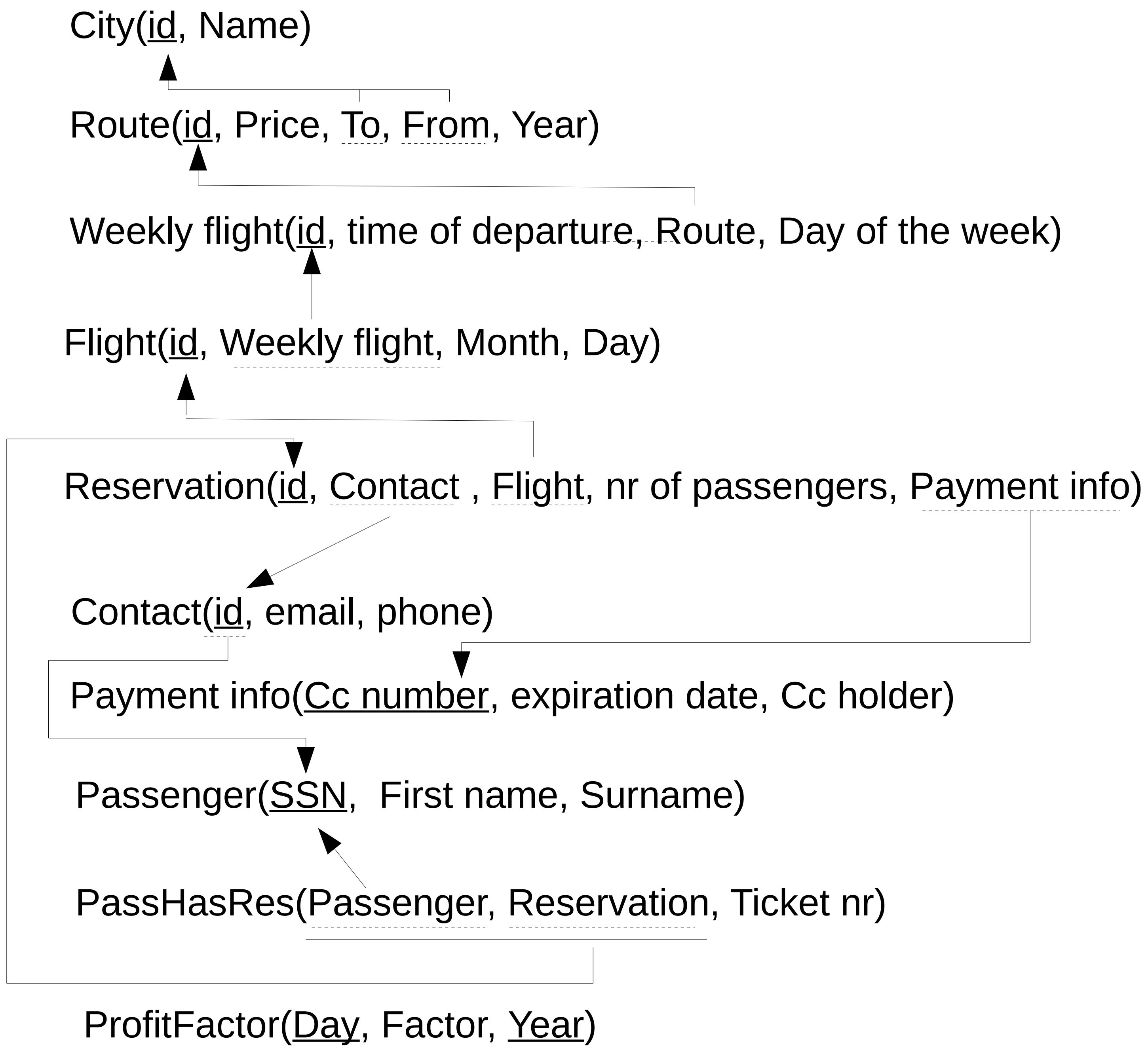

Year
Price
id
Route
From
To
N
N
1
1
1
City
id
Name
has
Day of the week
N
id
Weekly flight
1
Time of departure
Flies
Day
factor
Year
ProfitFactor
nr of passengers
id
N
Flight
1
Has
N
id
reservation
1
M
N
Date
Month
Day
Pays
1
Payment info
Has
Ticket nr
has
SSN
N
1
Passenger
Contact
Credit card number
Expiration date
Credit card holder
Name
First name
Surname
id
email
phone

Before normalisation



City(id, Name)

Route(id, Price, To, From, Year)

Weekly flight(id, time of departure, Route, Day of the week)

Flight(id, Weekly flight, Month, Day)

Reservation(id, Contact , Flight, nr of passengers, Payment info)

Contact(id, email, phone)

Payment info(Cc number, expiration date, Cc holder)

Passenger(SSN, First name, Surname)

PassHasRes(Passenger, Reservation, Ticket nr)

ProfitFactor(Day, Factor, Year)

The purpose of every table:

City:
The city table holds a name for a city and a unique id because there's a small risk that there will be two different cities sharing the same name. We use this table so we can avoid duplicating the city name in the route table.

Candidate keys: id
Primary key: id

Route:
This table contains the different routes that flights uses, used to avoid duplicating data when different flights use the same route. Stores info about arrival and destination city aswell as the cost for the specific route. The table also has a column that tells which year the route is used.

Candidate keys: id, (to, from, year)
Primary key: id

WeeklyFlight:
Stores a connection to a route, a weekday and a time of departure. Its purpose is to contain info for all the flights that might occur every week. The table also refers to the profitfactor table.

Candidate keys: id, (route, day of week, time of departure)
Primary key: id

ProfitFactor:
This table only has 3 columns. Day, factor and year are the values stored for each row. The purpose for this table is to avoid duplicate data and it also makes it possible to have different factors for different years.

Candidate keys: (Day, Year)
Primary key: (Day, Year)

Flights:
This table references a weekly flight and will be referenced to from other tables. The table also contains a month and a day.

Candidate keys: id, (weekly flight, month, day)
Primary key: id

Reservation:
Stores information about reservations that has been made. We also store information to a contact. Also contains number of passengers and a pointer to the payment info.

Candidate keys: id
Primary key: id

PaymentInfo:
Stores information about the credit card that paid for a booking.

Candidate keys: CC_number
Primary key: CC_number

Passenger:
Stores information about all passengers, each passenger will only be present once.

Candidate keys: SSN
Primary key: SSN

PassHasRes:
Stores the relation between reservations and passengers, it's needed since each passenger can have many reservations and a reservation can have many passengers. If a reservation has been paid for it will also contain the ticket number.

Candidate keys: (Passenger, Reservation)
Primary key: (Passenger, Reservation)

Contact:
Stores contact information.

Candidate keys: id
Primary key: id

We went through the normalization process and couldn't find any changes that was needed.

Lab 4 Project Answers
Marcus Sneitz(marsn336)
Erik Sneitz(erisn687)

4c)

To protect the data we could encrypt the information so it's not stored in plain text.

4e)

- It's more secure to keep all the logic server-side.
- It's possible for different types of applications to interact with the database.
- All logic is at one place and it's easy to maintain without having to think about what
framework and language the front-end uses.
- It's easier to scale in size.

9)

No, the reservation isn't visible in the other session because it's not commited to disc.
We can't modify the reservation since we can't see it at all.

To solve this issue we can manually COMMIT the changes directly after a procedure is done, this way
we will have the data on disc. If a procedure fails we can do a ROLLBACK which reverts all changes the procedure did.

10a)

If we call the create_reservation procedure at the same time from two different sessions and we check the available seats
we will get an overbooking since the sessions aren't aware of each others actions. We don't know that there are in fact no more seats left since the changes arent yet commited to disc and the session can interact with the same table at the same time.

10c)

To avoid these types of problem we can LOCK all the tables that we need in a procedure before we call it and then
release them afterwards, this way we don't allow procedures to read from a table that currently is write locked. This means
that one of the sessions has to wait for the other before it can proceed but it won't be possible to create overbookings.

```
/*
	Reset the db
*/
DROP DATABASE IF EXISTS brianair;
CREATE DATABASE brianair;
USE brianair;

/*
	Create tables
*/

CREATE TABLE City
(
	id INT NOT NULL AUTO_INCREMENT,
	name VARCHAR(32),
	PRIMARY KEY(id)
);

CREATE TABLE Route
(
	id INT NOT NULL AUTO_INCREMENT,
	year INT NOT NULL,
	price INT NOT NULL,
	start INT NOT NULL,
	destination INT NOT NULL,
	PRIMARY KEY(id),
	FOREIGN KEY(start) REFERENCES City(id),
	FOREIGN KEY(destination) REFERENCES City(id)
);

CREATE TABLE WeeklyFlight
(
	id INT NOT NULL AUTO_INCREMENT,
	departure_time TIME NOT NULL,
	weekday VARCHAR(32) NOT NULL,
	route INT NOT NULL,
	PRIMARY KEY(id),
	FOREIGN KEY(route) REFERENCES Route(id)
);

CREATE TABLE Flight
(
	id INT NOT NULL AUTO_INCREMENT,
	month INT NOT NULL,
	day INT NOT NULL,
	weekly_flight INT NOT NULL,
	PRIMARY KEY(id),
	FOREIGN KEY(weekly_flight) REFERENCES WeeklyFlight(id)
);

CREATE TABLE ProfitFactor
(
	day VARCHAR(32) NOT NULL,
	year INT NOT NULL,
	factor DOUBLE NOT NULL,
	PRIMARY KEY(day, year)
);

CREATE TABLE Passenger
(
	ssn BIGINT NOT NULL,
	fname VARCHAR(32) NOT NULL,
	lname VARCHAR(32) NOT NULL,
	PRIMARY KEY(ssn)
);

CREATE TABLE Contact
(
	id BIGINT NOT NULL,
	email VARCHAR(50) NOT NULL,
	phone BIGINT NOT NULL,
	PRIMARY KEY(id),
	FOREIGN KEY(id) REFERENCES Passenger(ssn)
);

CREATE TABLE PaymentInfo
(
	cc_number BIGINT NOT NULL,
	cc_holder VARCHAR(32) NOT NULL,
	expiration_day INT NOT NULL,
	expiration_month INT NOT NULL,
	PRIMARY KEY(cc_number)
);

CREATE TABLE Reservation
(
	id INT NOT NULL AUTO_INCREMENT,
	nr_of_passengers INT NOT NULL,
	price INT,
	flight INT NOT NULL,
	contact BIGINT,
	payment_info BIGINT,
	PRIMARY KEY(id),
	FOREIGN KEY(flight) REFERENCES Flight(id),
	FOREIGN KEY(contact) REFERENCES Contact(id),
	FOREIGN KEY(payment_info) REFERENCES PaymentInfo(cc_number)
);

CREATE TABLE Ticket
(
	reservation INT NOT NULL,
	passenger BIGINT NOT NULL,
	ticket VARCHAR(32) UNIQUE,
	PRIMARY KEY(reservation, passenger),
	FOREIGN KEY(reservation) REFERENCES Reservation(id),
	FOREIGN KEY(passenger) REFERENCES Passenger(ssn)
);

CREATE VIEW SearchView
AS

SELECT Flight.id, Flight.day, Flight.month, WeeklyFlight.departure_time, Route.year,
Departure.name AS Departure, Destination.name AS Destination
FROM Flight, WeeklyFlight, Route, City AS Departure, City AS Destination
WHERE Flight.weekly_flight = WeeklyFlight.id
AND WeeklyFlight.route = Route.id
AND Route.start = Departure.id
AND Route.destination = Destination.id;
	/*
SELECT Flight.id, Flight.day, Flight.month, WeeklyFlight.departure_time
FROM Flight
JOIN WeeklyFlight
ON Flight.weekly_flight = WeeklyFlight.id;*/

/*
	Setup procedures
*/

DELIMITER //

CREATE PROCEDURE create_reservation(flight INT, nr_of_passengers INT, OUT
reservation INT)
BEGIN
	IF (check_available_seats(flight) >= nr_of_passengers) THEN
	INSERT INTO Reservation(flight, nr_of_passengers)
	VALUES(flight, nr_of_passengers);
	SET reservation = LAST_INSERT_ID();
	ELSE
```

```
        SIGNAL SQLSTATE '13307'
        SET MESSAGE_TEXT = "Not enough available seats";
        END IF;

END //

CREATE PROCEDURE add_passenger(reservation_id INT, ssn_in BIGINT, fname
VARCHAR(32), lname VARCHAR(32), OUT passenger BIGINT)
BEGIN
        IF (SELECT COUNT(*) FROM Ticket WHERE reservation = reservation_id) <
(SELECT nr_of_passengers FROM Reservation WHERE id = reservation_id) THEN
        IF (SELECT COUNT(*) FROM Passenger WHERE ssn = ssn_in) = 0 THEN
        INSERT INTO Passenger(ssn, fname, lname)
        VALUES(ssn_in, fname, lname);
        END IF;
        SET passenger = ssn_in;
        INSERT INTO Ticket(reservation, passenger)
        VALUES(reservation_id, ssn_in);
        ELSE
        SIGNAL SQLSTATE '13307'
        SET MESSAGE_TEXT = "Cant add more passengers to this reservatiion.";
        END IF;
END //

CREATE PROCEDURE add_contact(reservation INT, passenger BIGINT, email_in
VARCHAR(50), phone_in BIGINT)
BEGIN
        IF (SELECT COUNT(*) FROM Contact WHERE id = passenger) = 0 THEN
        INSERT INTO Contact(id, email, phone)
        VALUES(passenger, email, phone);
        UPDATE Reservation
        SET contact = passenger
        WHERE id = reservation;
        ELSE
        UPDATE Reservation
        SET contact = passenger,
           email = email_in,
           phone = phone_in
        WHERE id = reservation;
        END IF;
END //

CREATE PROCEDURE add_payment(reservation_id INT, cc_number_in BIGINT, cc_holder
VARCHAR(32), expiration_day INT, expiration_month INT)
BEGIN
```

```
        IF (SELECT COUNT(*) FROM Ticket WHERE reservation = reservation_id) =
(SELECT nr_of_passengers FROM Reservation AS r1 WHERE id = reservation_id) AND
(SELECT contact FROM Reservation AS r2 WHERE id = reservation_id) IS NOT NULL
THEN
          SET @flight = (SELECT flight FROM Reservation AS r3 WHERE id =
reservation_id);
          IF (SELECT COUNT(*) FROM PaymentInfo WHERE cc_number = cc_number_in)
= 0 THEN
            INSERT INTO PaymentInfo(cc_number, cc_holder, expiration_day,
expiration_month)
            VALUES(cc_number_in, cc_holder, expiration_day, expiration_month);
            UPDATE Reservation
            SET payment_info = cc_number_in,
                 price = calc_price(@flight, nr_of_passengers)
            WHERE id = reservation_id;
          ELSE
            UPDATE Reservation
            SET payment_info = cc_number_in,
                 price = calc_price(@flight, nr_of_passengers)
            WHERE id = reservation_id;
          END IF;
        ELSE
          SIGNAL SQLSTATE '13307'
          SET MESSAGE_TEXT = "A reservation needs a contact aswell as the proper
amount of passengers.";
        END IF;
END //

CREATE PROCEDURE search(nr_of_passengers INT, day_in INT, month_in INT,
departure_in VARCHAR(32), destination_in VARCHAR(32))
BEGIN
        SELECT *, calc_price(id, nr_of_passengers) AS TotalSum,
check_available_seats(id) AS AvailableSeats
        FROM SearchView
        WHERE day = day_in
        AND month = month_in
        AND departure = departure_in
        AND destination = destination_in;
END //

DELIMITER ;

/*
        Setup triggers and functions
*/
```

```
DELIMITER //

CREATE FUNCTION check_available_seats(flight_id INT)
RETURNS INT
BEGIN
        RETURN(
                60 - (
                SELECT IFNULL(SUM(nr_of_passengers),0)
                FROM Reservation
                WHERE flight = flight_id)
        );
END //

CREATE FUNCTION occupied_seats(flight_id INT)
RETURNS INT
BEGIN
        RETURN(
                (
                SELECT COUNT(*)
                FROM Ticket
                WHERE reservation IN (SELECT r4.id FROM Reservation AS r4 WHERE
r4.flight = flight_id)
                AND ticket IS NOT NULL
                )
        );
END //

CREATE FUNCTION calc_price(flight_id INT, nr_of_passengers INT)
RETURNS INT
BEGIN
        RETURN(
                (SELECT price FROM Route AS route1 WHERE id =
                        (SELECT route FROM WeeklyFlight AS wf1 WHERE id =
                                (SELECT weekly_flight FROM Flight AS f1 WHERE id =
flight_id)))
                *(SELECT factor FROM ProfitFactor WHERE day LIKE
                         (SELECT weekday FROM WeeklyFlight AS wf2 WHERE id =
                                 (SELECT weekly_flight FROM Flight AS f2 WHERE id =
flight_id))
                                AND year = (
                                   SELECT year FROM Route route2 WHERE id = (
                                          SELECT route FROM WeeklyFlight AS wf3 WHERE
id = (
                                                  SELECT weekly_flight FROM Flight AS f3
WHERE id = flight_id))))
              *(occupied_seats(flight_id)+1)/60
```

```
                *nr_of_passengers
        );
END //

CREATE FUNCTION generate_unique_ticket()
RETURNS VARCHAR(32)
BEGIN
        RETURN(
                (SELECT SUBSTRING(MD5(RAND()) FROM 1 FOR 32))
        );
END //

CREATE TRIGGER on_payment
AFTER UPDATE
ON Reservation
FOR EACH ROW
BEGIN
        IF NEW.payment_info THEN
          UPDATE Ticket
          SET ticket = generate_unique_ticket()
          WHERE reservation = NEW.id;
        END IF;
END //

DELIMITER ;

/*
        Insert some data
*/

INSERT INTO ProfitFactor(day, year, factor)
VALUES
        ("Monday", 2015, 1),
        ("Tuesday", 2015, 1),
        ("Wednesday", 2015, 1),
        ("Thursday", 2015, 1),
        ("Friday", 2015, 1.5),
        ("Saturday", 2015, 2),
        ("Sunday", 2015, 1.5),
        ("Monday", 2016, 1.2),
        ("Tuesday", 2016, 1.2),
        ("Wednesday", 2016, 1.2),
        ("Thursday", 2016, 1.2),
        ("Friday", 2016, 1.8),
        ("Saturday", 2016, 2.3),
        ("Sunday", 2016, 2);
```

```
INSERT INTO City(name)
VALUES
	("Linköping"),
	("Jönköping"),
	("Stockholm"),
	("Malmö"),
	("Göteborg"),
	("Uppsala");

INSERT INTO Route(year, price, start, destination)
VALUES
	(
		2015,
		800,
		(SELECT id FROM City WHERE name LIKE "Jönköping"),
		(SELECT id FROM City WHERE name LIKE "Linköping")
	),
	(
		2015,
		1000,
		(SELECT id FROM City WHERE name LIKE "Uppsala"),
		(SELECT id FROM City WHERE name LIKE "Linköping")
	),
	(
		2016,
		800,
		(SELECT id FROM City WHERE name LIKE "Göteborg"),
		(SELECT id FROM City WHERE name LIKE "Jönköping")
	);

INSERT INTO WeeklyFlight(departure_time, weekday, route)
VALUES
	(
		"16:45:00",
		"Monday",
		(SELECT id FROM Route WHERE start = (SELECT id FROM City WHERE
name LIKE "Jönköping")
		AND destination = (SELECT id FROM City WHERE name LIKE "Linköping"))
	),
	(
		"23:12:00",
		"Thursday",
		(SELECT id FROM Route WHERE start = (SELECT id FROM City WHERE
name LIKE "Uppsala")
		AND destination = (SELECT id FROM City WHERE name LIKE "Linköping"))
	),
	(
		"08:00:00",
		"Friday",
		(SELECT id FROM Route WHERE start = (SELECT id FROM City WHERE
name LIKE "Jönköping")
		AND destination = (SELECT id FROM City WHERE name LIKE "Linköping"))
	),
	(
		"14:30:00",
		"Saturday",
		(SELECT id FROM Route WHERE start = (SELECT id FROM City WHERE
name LIKE "Göteborg")
		AND destination = (SELECT id FROM City WHERE name LIKE "Jönköping"))
	),
	(
		"13:37:00",
		"Sunday",
		(SELECT id FROM Route WHERE start = (SELECT id FROM City WHERE
name LIKE "Uppsala")
		AND destination = (SELECT id FROM City WHERE name LIKE "Linköping"))
	);

INSERT INTO Flight(month, day, weekly_flight)
VALUES
	(
		5,
		31,
		1
	),
	(
		6,
		3,
		2
	),
	(
		6,
		8,
		3
	),
	(
		6,
		13,
		4
	),
	(
        6,
        16,
        5
);
```

```
SET AUTOCOMMIT = 0;
START TRANSACTION;
LOCK TABLES Reservation WRITE;
CALL create_reservation(1, 2, @res);
UNLOCK TABLES;
COMMIT;
START TRANSACTION;
LOCK TABLES Passenger WRITE, Ticket WRITE, Reservation READ;
CALL add_passenger(@res ,9407021337,"Heriku","Sneitz", @pass);
CALL add_passenger(@res,9205221337,"Mac","Sneitz", @pass);
CALL add_passenger(@res,9205221338,"Fail","Failure", @pass);
UNLOCK TABLES;
COMMIT;

START TRANSACTION;
LOCK TABLES Reservation WRITE;
CALL create_reservation(1, 2, @res);
UNLOCK TABLES;
COMMIT;

START TRANSACTION;
LOCK TABLES
Passenger WRITE,
Ticket WRITE,
Reservation WRITE,
PaymentInfo WRITE,
Contact WRITE;
CALL add_passenger(@res ,9407020016,"Erik","Sneitz", @pass);
CALL add_passenger(@res,9205221337,"Mac","Sneitz", @pass);
CALL add_contact(@res, @pass, "lol@lol.com", 1337);
UNLOCK TABLES;
COMMIT;
START TRANSACTION;
LOCK TABLES
PaymentInfo WRITE,
ProfitFactor READ,
Route AS route1 READ,
Route AS route2 READ,
Reservation WRITE,
Reservation AS r1 WRITE,
Reservation AS r2 WRITE,
Reservation AS r3 WRITE,
Reservation AS r4 WRITE,
WeeklyFlight READ,
WeeklyFlight AS wf1 READ,
WeeklyFlight AS wf2 READ,
WeeklyFlight AS wf3 READ,
Flight AS f1 READ,
Flight AS f2 READ,
Flight AS f3 READ,
Ticket WRITE;
CALL add_payment(@res, 1337127688103333, "Erik Sneitz", 5, 12);
UNLOCK TABLES;
COMMIT;

START TRANSACTION;
LOCK TABLES
Reservation WRITE;
CALL create_reservation(1, 2, @res);
UNLOCK TABLES;
COMMIT;

START TRANSACTION;
LOCK TABLES
Passenger WRITE,
Ticket WRITE,
Reservation WRITE,
PaymentInfo WRITE,
Contact WRITE;
CALL add_passenger(@res ,94070201116,"Test","Sneitz", @pass);
CALL add_passenger(@res,920522123,"Test2","Sneitz", @pass);
CALL add_contact(@res, @pass, "lol@lol.com", 1337);
UNLOCK TABLES;
COMMIT;

START TRANSACTION;
LOCK TABLES
PaymentInfo WRITE,
ProfitFactor READ,
Route AS route1 READ,
Route AS route2 READ,
Reservation WRITE,
Reservation AS r1 WRITE,
Reservation AS r2 WRITE,
Reservation AS r3 WRITE,
Reservation AS r4 WRITE,
WeeklyFlight READ,
WeeklyFlight AS wf1 READ,
WeeklyFlight AS wf2 READ,
WeeklyFlight AS wf3 READ,
Flight AS f1 READ,
Flight AS f2 READ,
Flight AS f3 READ,
Ticket WRITE;
CALL add_payment(@res, 1337127688103333, "Erik Sneitz", 5, 12);
UNLOCK TABLES;
COMMIT;

START TRANSACTION;
LOCK TABLES
Reservation WRITE;
CALL create_reservation(1, 30, @res);
UNLOCK TABLES;
COMMIT;

START TRANSACTION;
LOCK TABLES
Reservation WRITE;
CALL create_reservation(1, 30, @res);
UNLOCK TABLES;
COMMIT;

START TRANSACTION;
LOCK TABLES
SearchView READ,
ProfitFactor READ,
Reservation READ,
Reservation AS r4 WRITE,
WeeklyFlight AS wf1 READ,
WeeklyFlight AS wf2 READ,
WeeklyFlight AS wf3 READ,
Flight AS f1 READ,
Flight AS f2 READ,
Flight AS f3 READ,
Route AS route1 READ,
Route AS route2 READ;
CALL search(5,31,5,"Jönköping","Linköping");
UNLOCK TABLES;
COMMIT;

START TRANSACTION;
LOCK TABLES Reservation READ;
SET @available_seats = check_available_seats(1);
SELECT @available_seats;
UNLOCK TABLES;
COMMIT;

START TRANSACTION;
```

```
LOCK TABLES
Reservation AS r4 READ,
Ticket READ;
SET @booked_seats = occupied_seats(1);
SELECT @booked_seats;
UNLOCK TABLES;
COMMIT;

START TRANSACTION;
LOCK TABLES
ProfitFactor READ,
Reservation AS r4 WRITE,
WeeklyFlight AS wf1 READ,
WeeklyFlight AS wf2 READ,
WeeklyFlight AS wf3 READ,
Flight AS f1 READ,
Flight AS f2 READ,
Flight AS f3 READ,
Route AS route1 READ,
Route AS route2 READ;
SET @price = calc_price(1,3);
SELECT @price;
UNLOCK TABLES;
COMMIT;
```