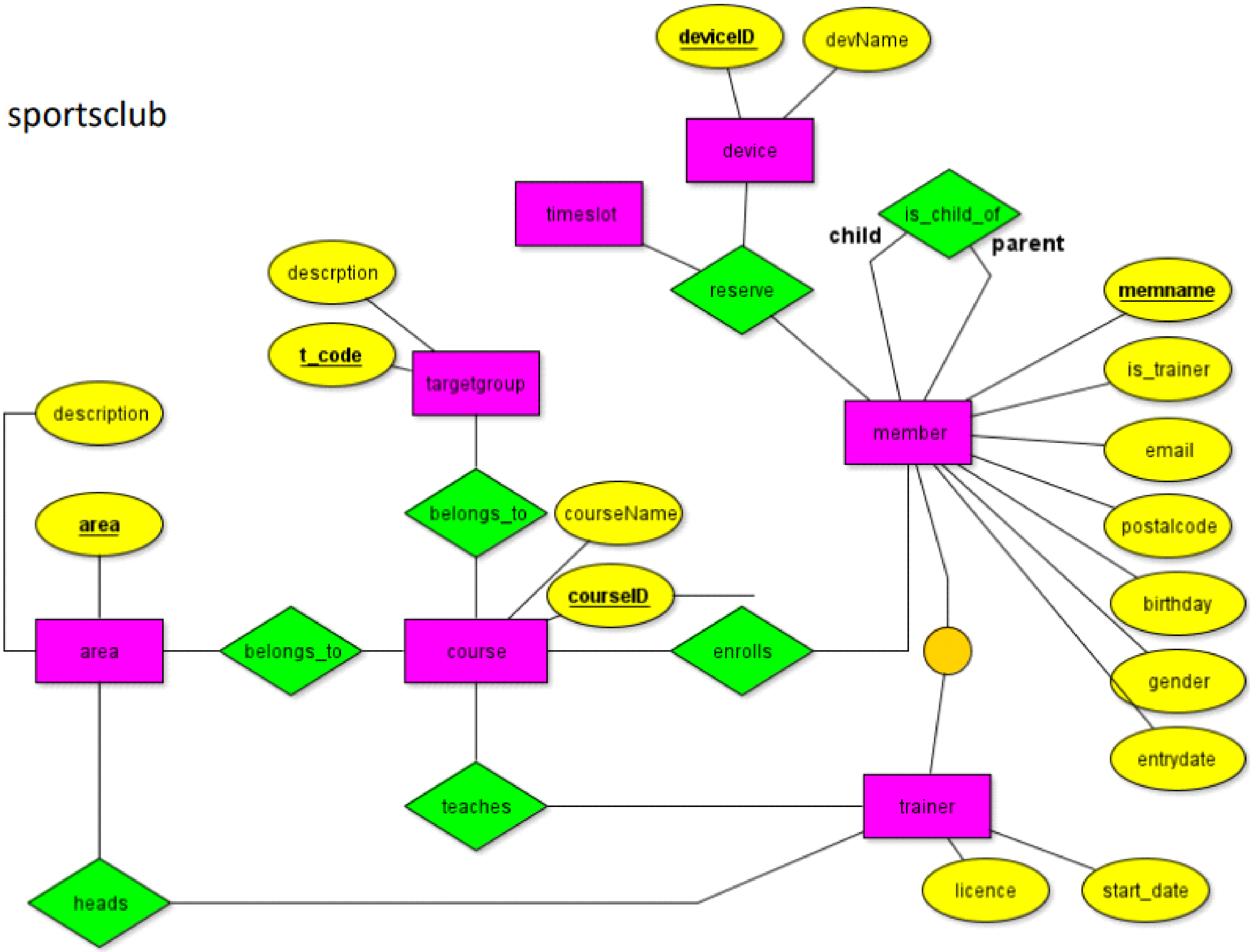
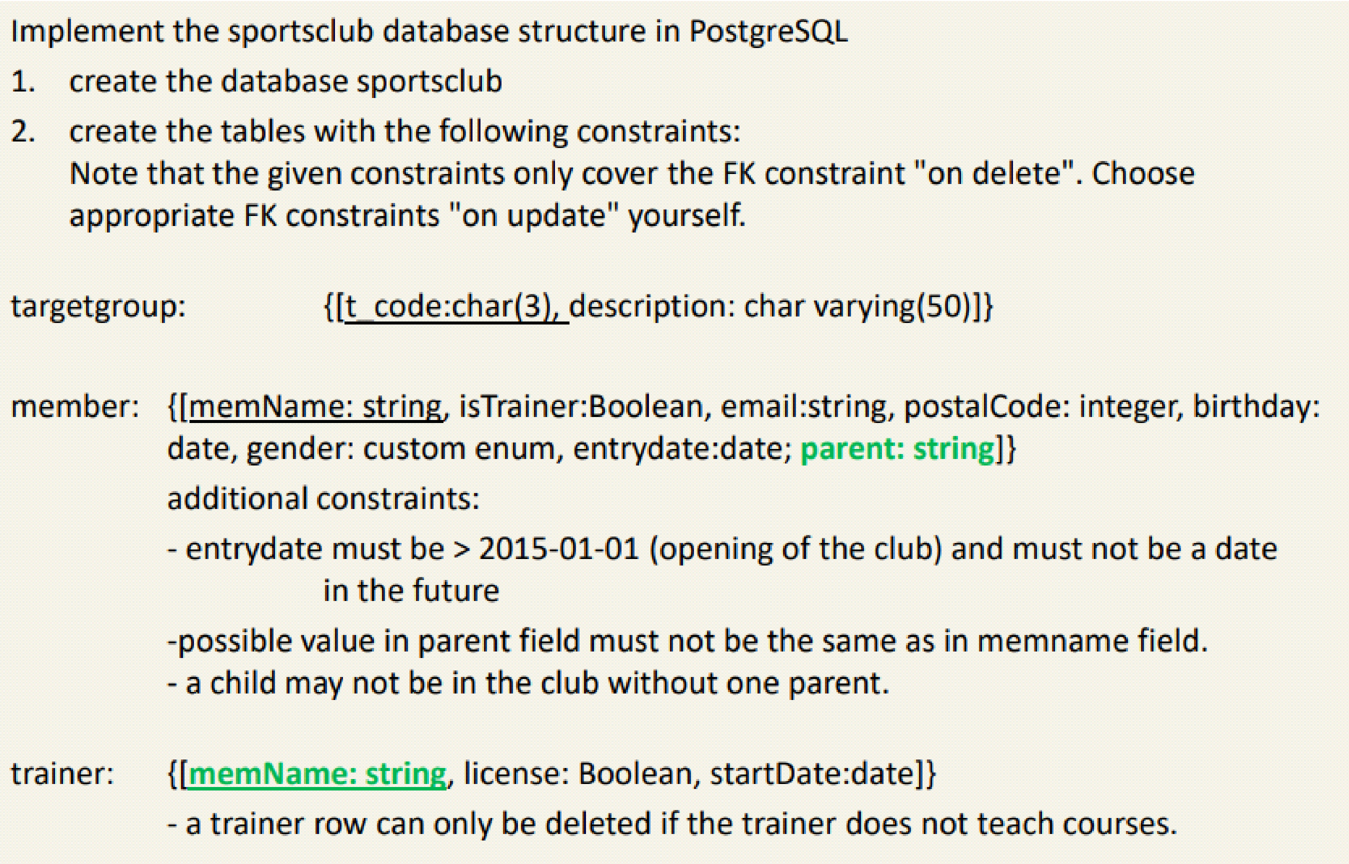
HW3

Thursday, October 24, 2024 5:44 PM





Implement the sportsclub database structure in PostgreSQL

1. create the database sportsclub
2. create the tables with the following constraints:
   Note that the given constraints only cover the FK constraint "on delete". Choose appropriate FK constraints "on update" yourself.

targetgroup: {[t_code:char(3), description: char varying(50)]}

member: {[memName: string, isTrainer:Boolean, email:string, postalCode: integer, birthday: date, gender: custom enum, entrydate:date; **parent: string**]}

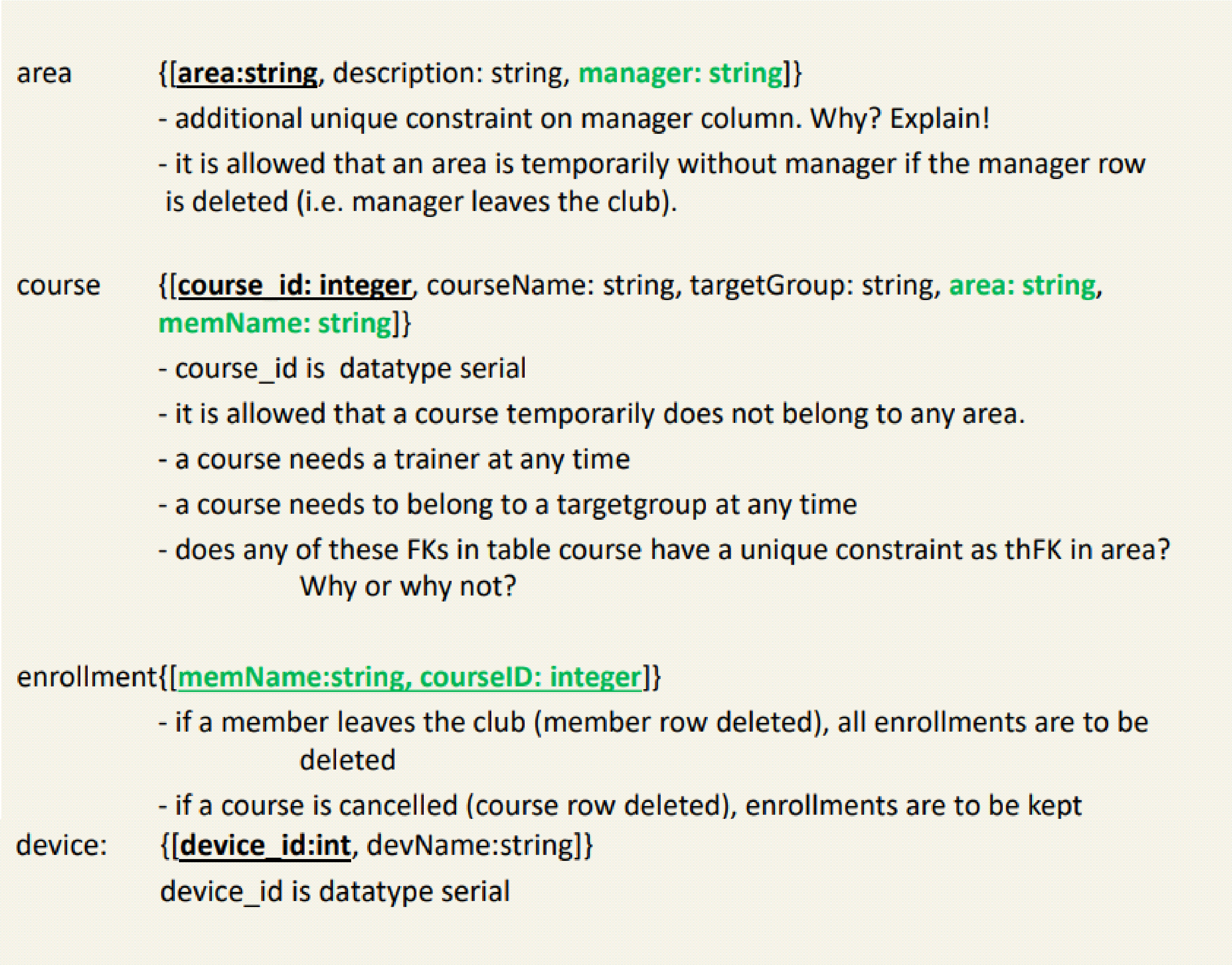
additional constraints:

- entrydate must be > 2015-01-01 (opening of the club) and must not be a date in the future

-possible value in parent field must not be the same as in memname field.
- a child may not be in the club without one parent.

trainer: {[**memName: string**, license: Boolean, startDate:date]}

- a trainer row can only be deleted if the trainer does not teach courses.



area {[**area:string**, description: string, **manager: string**]}

- additional unique constraint on manager column. Why? Explain!
- it is allowed that an area is temporarily without manager if the manager row is deleted (i.e. manager leaves the club).

course {[**course_id: integer**, courseName: string, targetGroup: string, **area: string**, **memName: string**]}

- course_id is datatype serial
- it is allowed that a course temporarily does not belong to any area.
- a course needs a trainer at any time
- a course needs to belong to a targetgroup at any time
- does any of these FKs in table course have a unique constraint as thFK in area? Why or why not?

enrollment{[**memName:string, courseID: integer**]}

- if a member leaves the club (member row deleted), all enrollments are to be deleted
- if a course is cancelled (course row deleted), enrollments are to be kept

device: {[**device_id:int**, devName:string]}

device_id is datatype serial

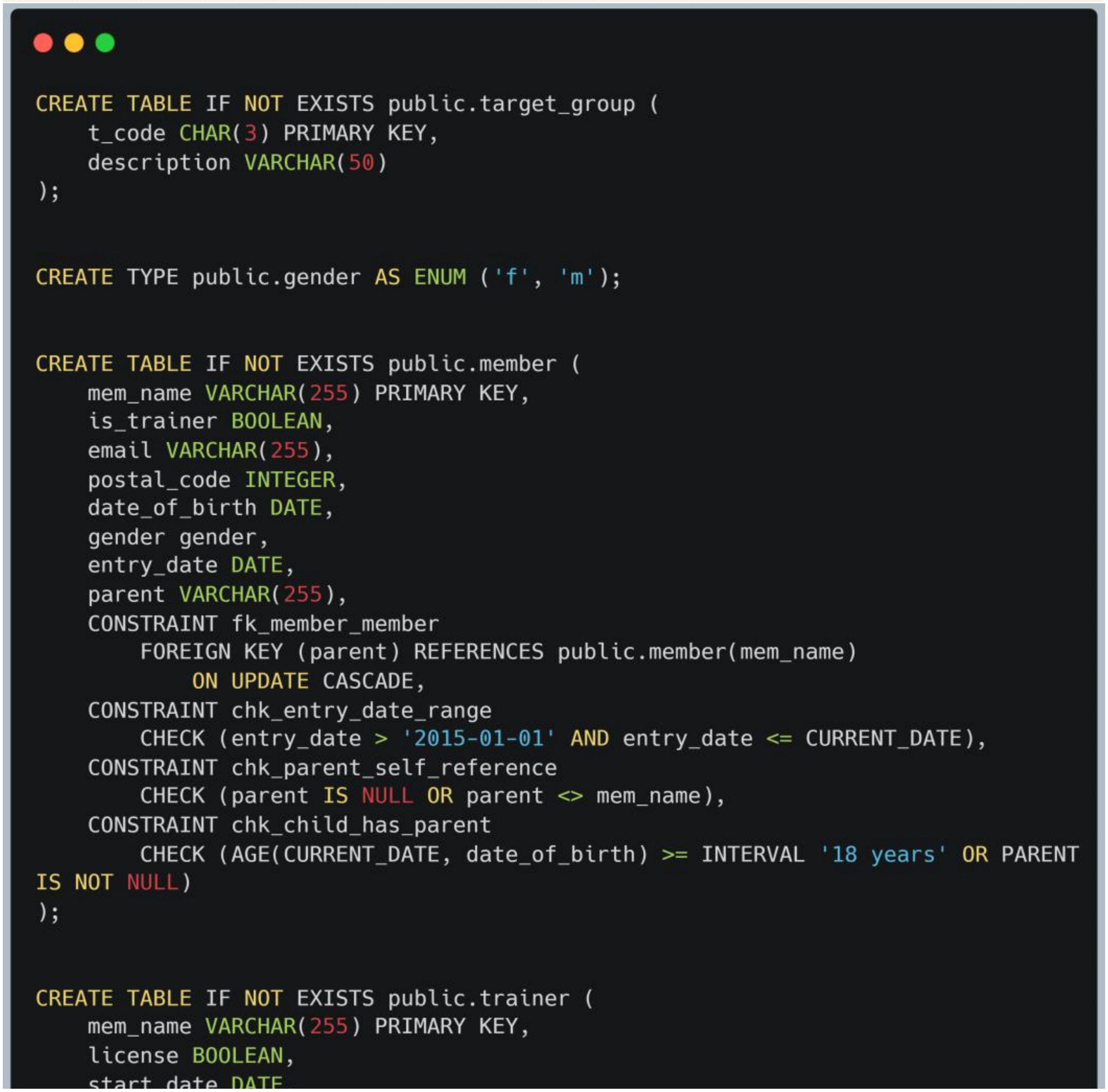
device: {[**device_id:int**, devName:string]}

device_id is datatype serial

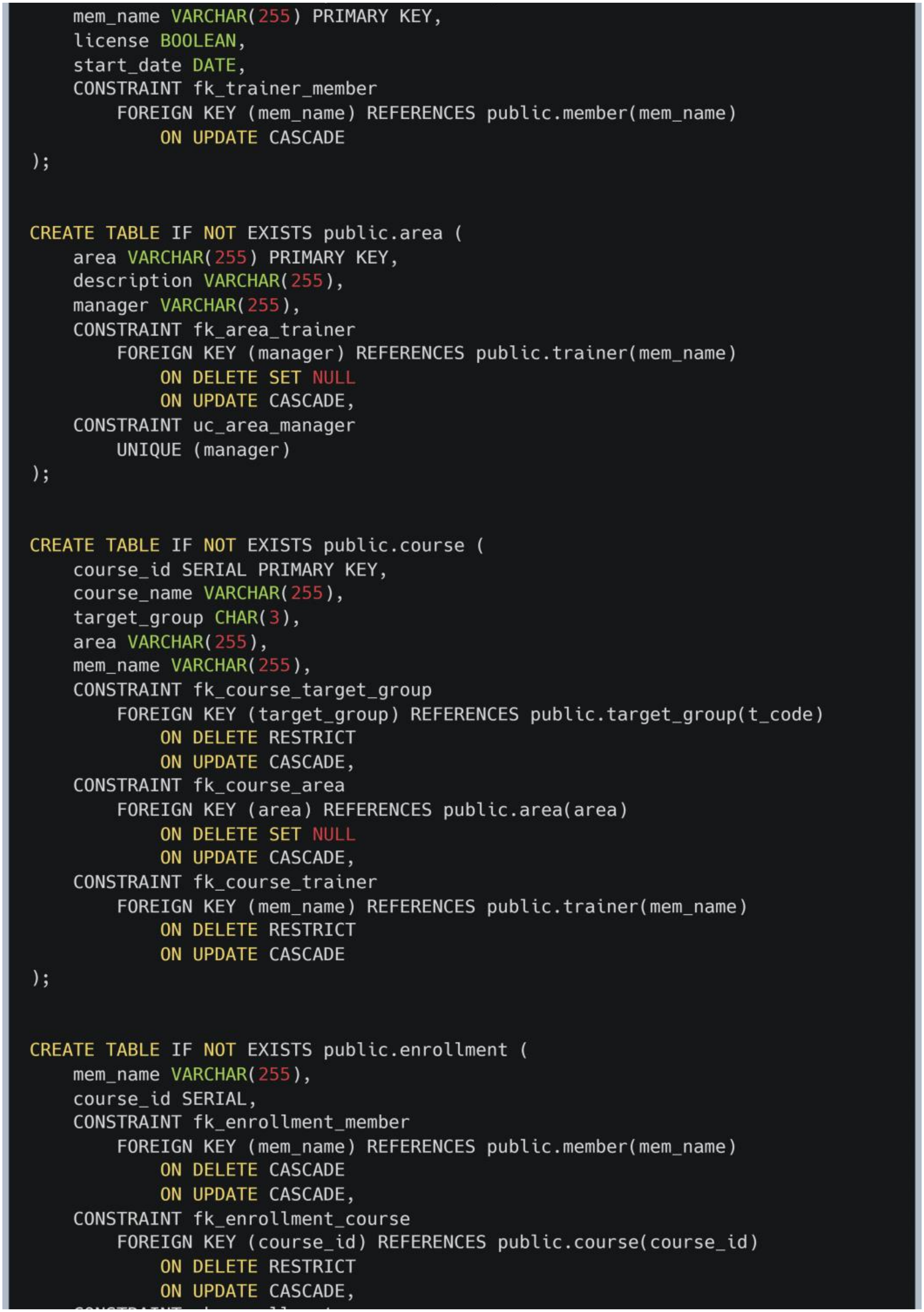
reservation: {[**timeslot:timestamp, memName:string, device_id:int**]}

- with additional unique constraint on {timeslot,, device_id}
- if a member row is deleted (member leaves the club), all the rreservations of the member need to be deleted
- if a device is not available anymore, the reservations of this device need to stay

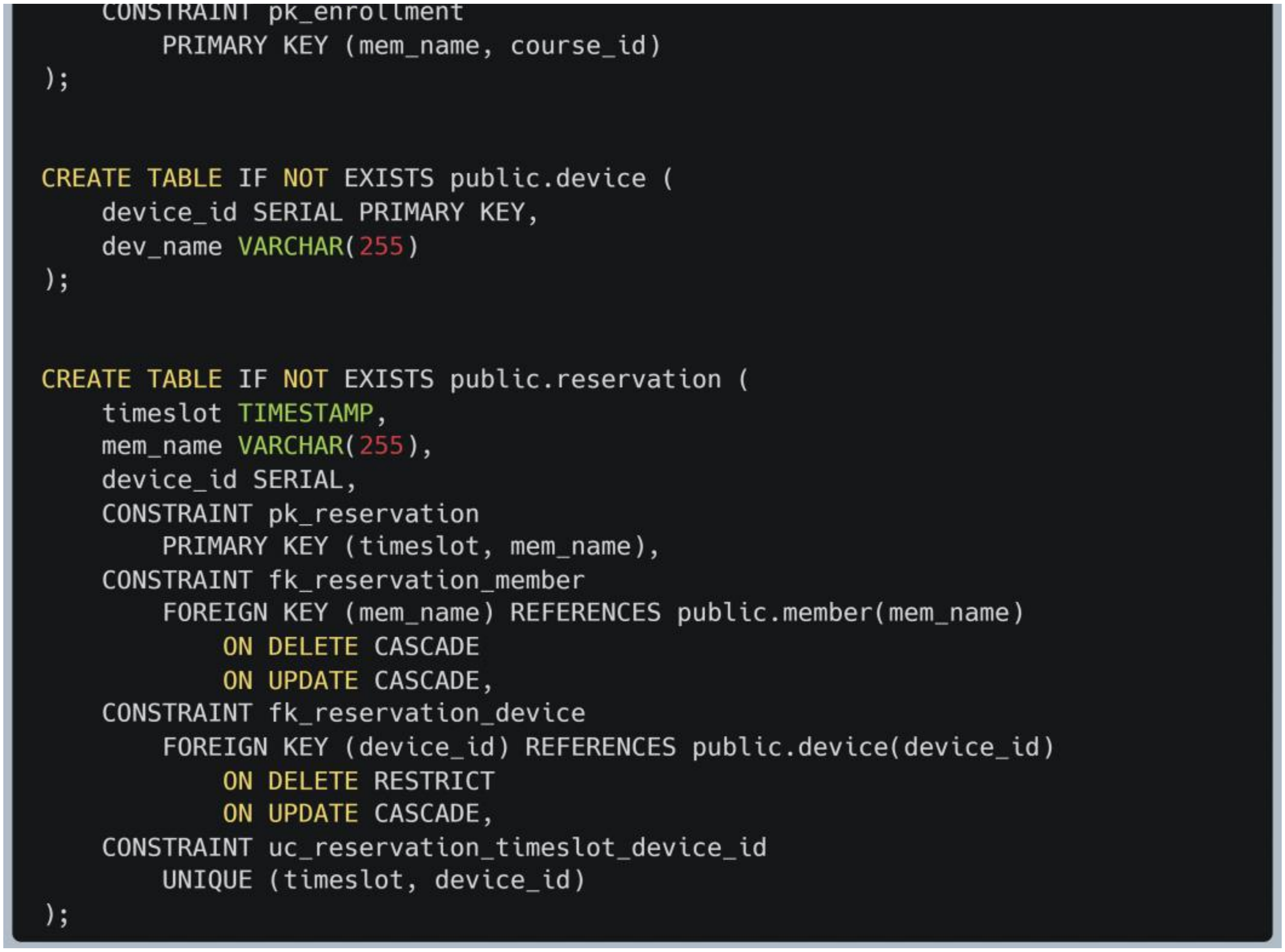
Backup the schema of your database!



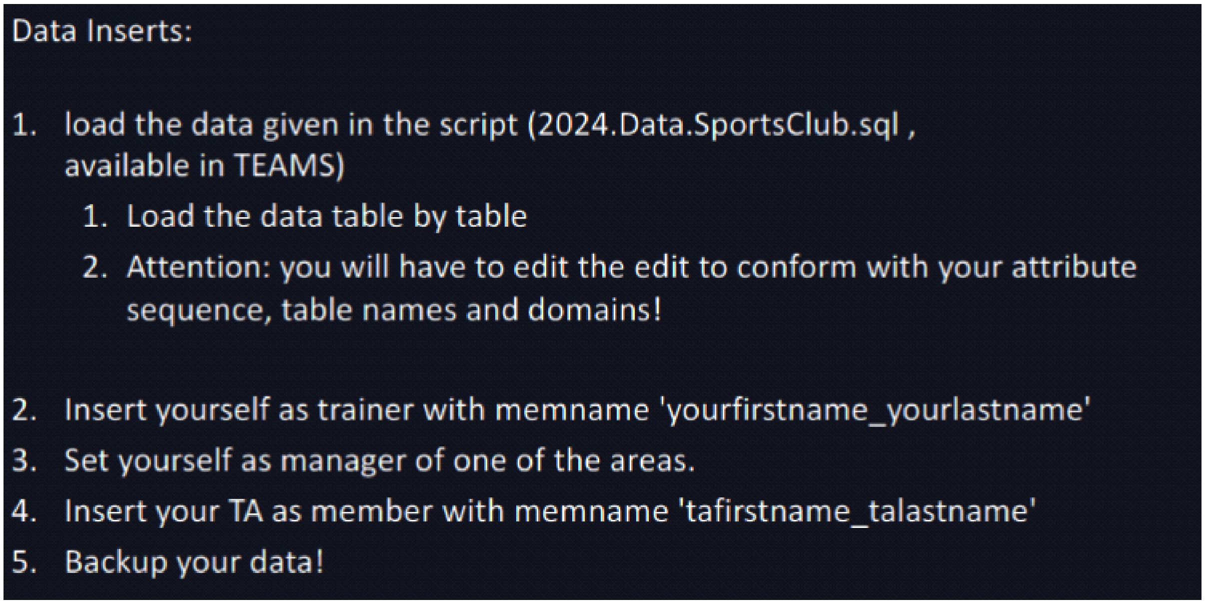
```
CREATE TABLE IF NOT EXISTS public.target_group (
    t_code CHAR(3) PRIMARY KEY,
    description VARCHAR(50)
);


CREATE TYPE public.gender AS ENUM ('f', 'm');


CREATE TABLE IF NOT EXISTS public.member (
    mem_name VARCHAR(255) PRIMARY KEY,
    is_trainer BOOLEAN,
    email VARCHAR(255),
    postal_code INTEGER,
    date_of_birth DATE,
    gender gender,
    entry_date DATE,
    parent VARCHAR(255),
    CONSTRAINT fk_member_member
        FOREIGN KEY (parent) REFERENCES public.member(mem_name)
            ON UPDATE CASCADE,
    CONSTRAINT chk_entry_date_range
        CHECK (entry_date > '2015-01-01' AND entry_date <= CURRENT_DATE),
    CONSTRAINT chk_parent_self_reference
        CHECK (parent IS NULL OR parent <> mem_name),
    CONSTRAINT chk_child_has_parent
        CHECK (AGE(CURRENT_DATE, date_of_birth) >= INTERVAL '18 years' OR PARENT
IS NOT NULL)
);


CREATE TABLE IF NOT EXISTS public.trainer (
    mem_name VARCHAR(255) PRIMARY KEY,
    license BOOLEAN,
    start_date DATE,
```



```sql
    mem_name VARCHAR(255) PRIMARY KEY,
    license BOOLEAN,
    start_date DATE,
    CONSTRAINT fk_trainer_member
        FOREIGN KEY (mem_name) REFERENCES public.member(mem_name)
            ON UPDATE CASCADE
);


CREATE TABLE IF NOT EXISTS public.area (
    area VARCHAR(255) PRIMARY KEY,
    description VARCHAR(255),
    manager VARCHAR(255),
    CONSTRAINT fk_area_trainer
        FOREIGN KEY (manager) REFERENCES public.trainer(mem_name)
            ON DELETE SET NULL
            ON UPDATE CASCADE,
    CONSTRAINT uc_area_manager
        UNIQUE (manager)
);


CREATE TABLE IF NOT EXISTS public.course (
    course_id SERIAL PRIMARY KEY,
    course_name VARCHAR(255),
    target_group CHAR(3),
    area VARCHAR(255),
    mem_name VARCHAR(255),
    CONSTRAINT fk_course_target_group
        FOREIGN KEY (target_group) REFERENCES public.target_group(t_code)
            ON DELETE RESTRICT
            ON UPDATE CASCADE,
    CONSTRAINT fk_course_area
        FOREIGN KEY (area) REFERENCES public.area(area)
            ON DELETE SET NULL
            ON UPDATE CASCADE,
    CONSTRAINT fk_course_trainer
        FOREIGN KEY (mem_name) REFERENCES public.trainer(mem_name)
            ON DELETE RESTRICT
            ON UPDATE CASCADE
);


CREATE TABLE IF NOT EXISTS public.enrollment (
    mem_name VARCHAR(255),
    course_id SERIAL,
    CONSTRAINT fk_enrollment_member
        FOREIGN KEY (mem_name) REFERENCES public.member(mem_name)
            ON DELETE CASCADE
            ON UPDATE CASCADE,
    CONSTRAINT fk_enrollment_course
        FOREIGN KEY (course_id) REFERENCES public.course(course_id)
            ON DELETE RESTRICT
            ON UPDATE CASCADE,
```



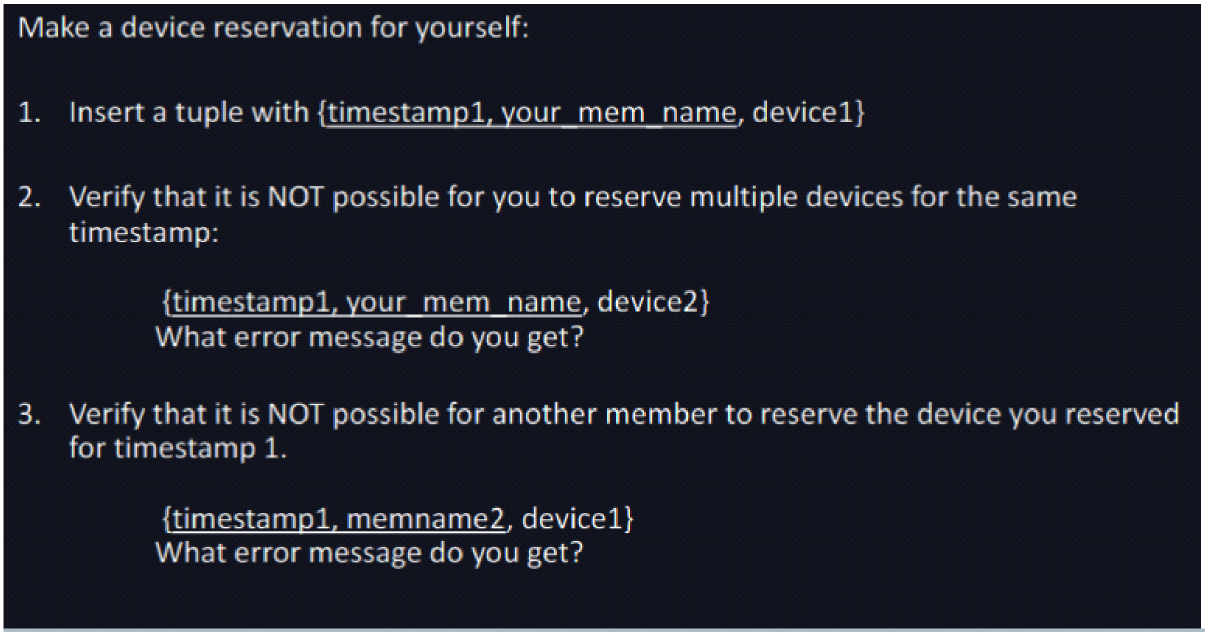
```
    CONSTRAINT pk_enrollment
        PRIMARY KEY (mem_name, course_id)
);


CREATE TABLE IF NOT EXISTS public.device (
    device_id SERIAL PRIMARY KEY,
    dev_name VARCHAR(255)
);


CREATE TABLE IF NOT EXISTS public.reservation (
    timeslot TIMESTAMP,
    mem_name VARCHAR(255),
    device_id SERIAL,
    CONSTRAINT pk_reservation
        PRIMARY KEY (timeslot, mem_name),
    CONSTRAINT fk_reservation_member
        FOREIGN KEY (mem_name) REFERENCES public.member(mem_name)
            ON DELETE CASCADE
            ON UPDATE CASCADE,
    CONSTRAINT fk_reservation_device
        FOREIGN KEY (device_id) REFERENCES public.device(device_id)
            ON DELETE RESTRICT
            ON UPDATE CASCADE,
    CONSTRAINT uc_reservation_timeslot_device_id
        UNIQUE (timeslot, device_id)
);
```



Data Inserts:

1. load the data given in the script (2024.Data.SportsClub.sql , available in TEAMS)
    1. Load the data table by table
    2. Attention: you will have to edit the edit to conform with your attribute sequence, table names and domains!

2. Insert yourself as trainer with memname 'yourfirstname_yourlastname'
3. Set yourself as manager of one of the areas.
4. Insert your TA as member with memname 'tafirstname_talastname'
5. Backup your data!



Make a device reservation for yourself:

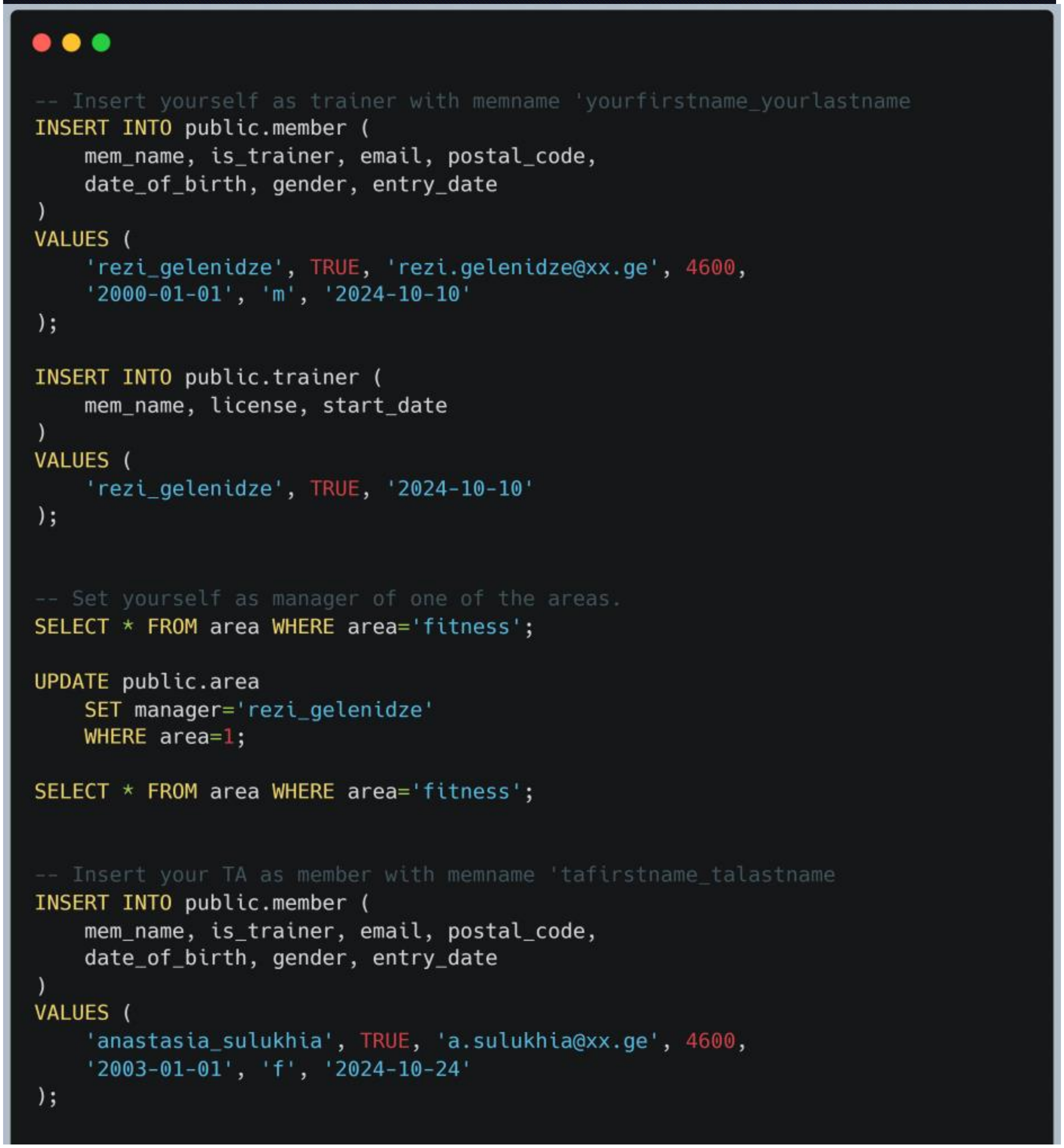
1. Insert a tuple with {timestamp1, your_mem_name, device1}

2. Verify that it is NOT possible for you to reserve multiple devices for the same timestamp:

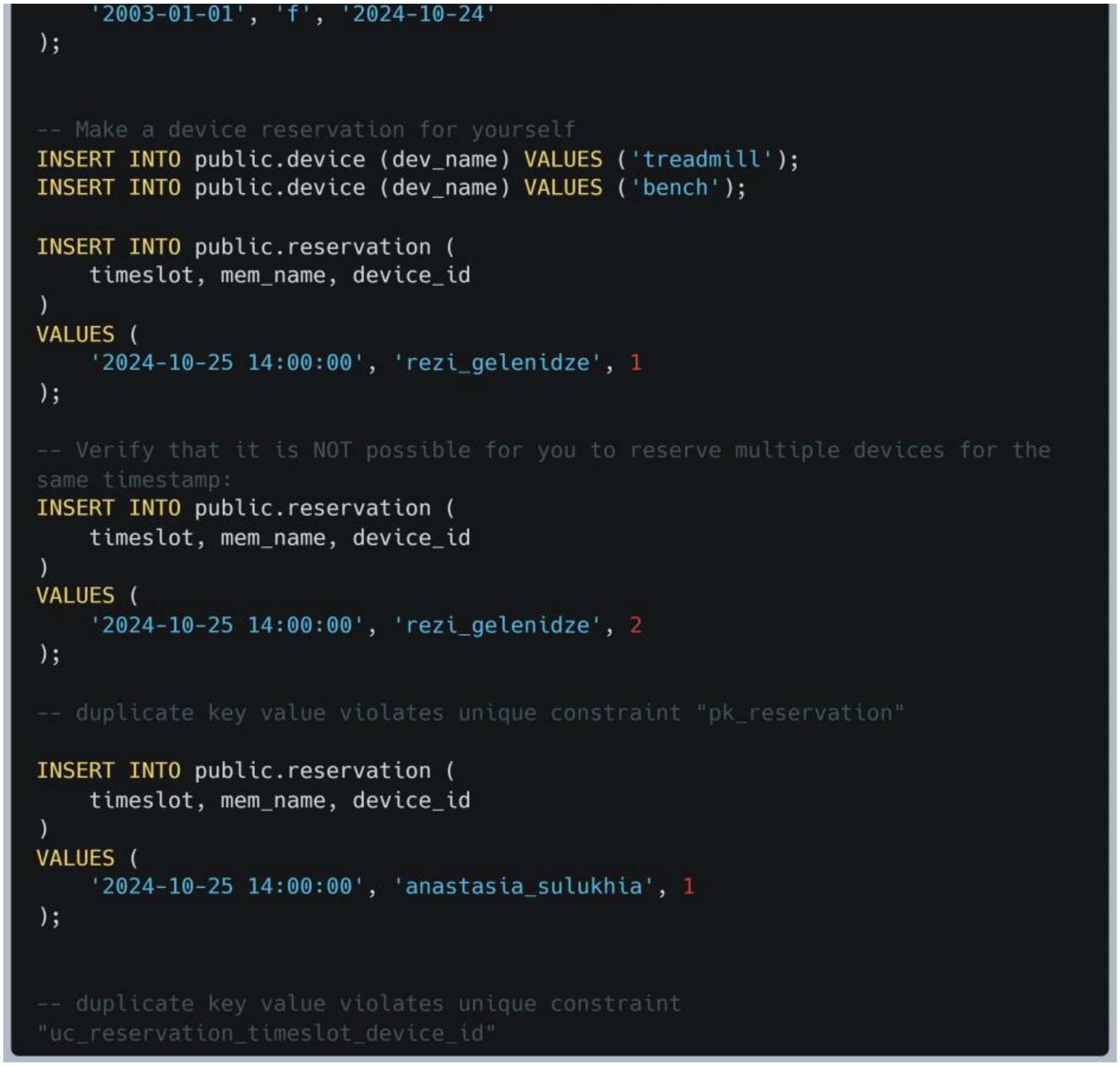
   {timestamp1, your_mem_name, device2}
   What error message do you get?

3. Verify that it is NOT possible for another member to reserve the device you reserved for timestamp 1.

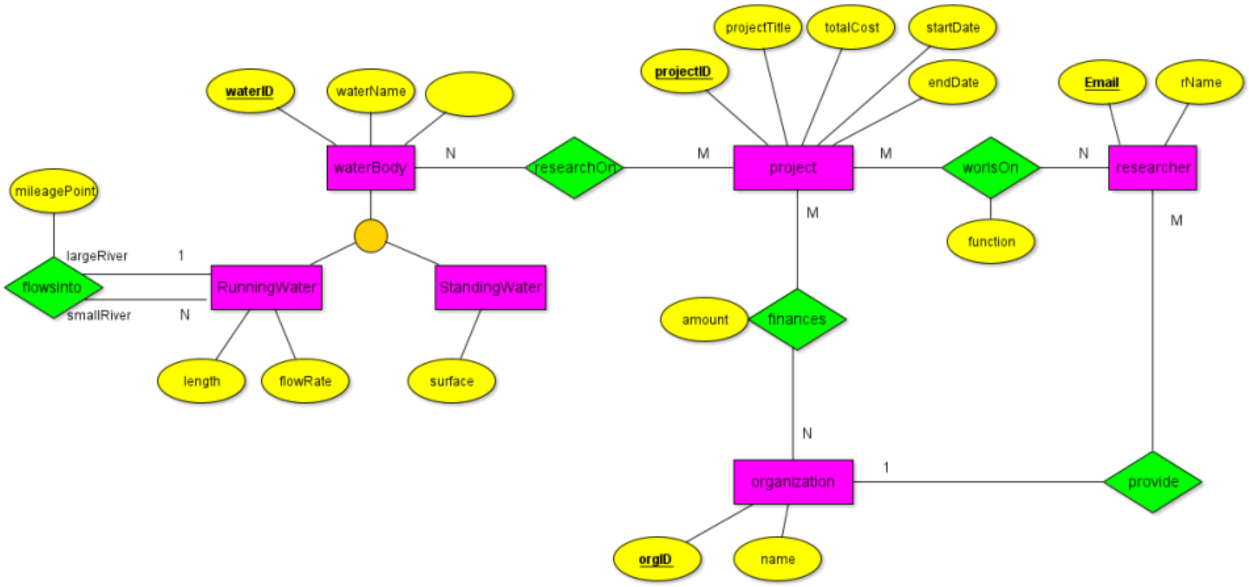
   {timestamp1, memname2, device1}
   What error message do you get?



```
-- Insert yourself as trainer with memname 'yourfirstname_yourlastname
INSERT INTO public.member (
    mem_name, is_trainer, email, postal_code,
    date_of_birth, gender, entry_date
)
VALUES (
    'rezi_gelenidze', TRUE, 'rezi.gelenidze@xx.ge', 4600,
    '2000-01-01', 'm', '2024-10-10'
);

INSERT INTO public.trainer (
    mem_name, license, start_date
)
VALUES (
    'rezi_gelenidze', TRUE, '2024-10-10'
);


-- Set yourself as manager of one of the areas.
SELECT * FROM area WHERE area='fitness';

UPDATE public.area
    SET manager='rezi_gelenidze'
    WHERE area=1;

SELECT * FROM area WHERE area='fitness';


-- Insert your TA as member with memname 'tafirstname_talastname
INSERT INTO public.member (
    mem_name, is_trainer, email, postal_code,
    date_of_birth, gender, entry_date
)
VALUES (
    'anastasia_sulukhia', TRUE, 'a.sulukhia@xx.ge', 4600,
    '2003-01-01', 'f', '2024-10-24'
);
```



```
    '2003-01-01', 'f', '2024-10-24'
);


-- Make a device reservation for yourself
INSERT INTO public.device (dev_name) VALUES ('treadmill');
INSERT INTO public.device (dev_name) VALUES ('bench');

INSERT INTO public.reservation (
    timeslot, mem_name, device_id
)
VALUES (
    '2024-10-25 14:00:00', 'rezi_gelenidze', 1
);

-- Verify that it is NOT possible for you to reserve multiple devices for the
same timestamp:
INSERT INTO public.reservation (
    timeslot, mem_name, device_id
)
VALUES (
    '2024-10-25 14:00:00', 'rezi_gelenidze', 2
);

-- duplicate key value violates unique constraint "pk_reservation"

INSERT INTO public.reservation (
    timeslot, mem_name, device_id
)
VALUES (
    '2024-10-25 14:00:00', 'anastasia_sulukhia', 1
);


-- duplicate key value violates unique constraint
"uc_reservation_timeslot_device_id"
```



Map the waterbaody ER model into a relational schema.
Use horizontal partitioning to map the generalization.
How many final relations do you get?
Are there FKs that need to be unique?

standing_water: {[water_id: INTEGER, water_name: VARCHAR, surface: INTEGER]}

running_water: {[water_id: INTEGER, water_name: VARCHAR, length: DECIMAL, flow_rate: DECIMAL, flows_into: INTEGER, mileage_point: INTEGER]}

organization: {[org_id: INTEGER, name: VARCHAR]}

project: {[project_id: INTEGER, project_title: VARCHAR, total_cost: INTEGER, start_date: DATE, end_date: DATE]}

researcher: {[email: VARCHAR, r_name: VARCHAR, provides: INTEGER]}

organization_finances: {[org_id: INTEGER, project_id: INTEGER, amount: INTEGER]}

research_on_standing_water: {[project_id: INTEGER, water_id: INTEGER]}

research_on_running_water: {[project_id: INTEGER, water_id: INTEGER]}

researcher_works_on: {[email: VARCHAR, project_id: INTEGER, function: VARCHAR]}

3. We got 9 relations (8 possible if we merge horizontal identical relations of research_on)
4. We don't have exact requirements to infer constraints for that