FINAL PROJECT PROPOSAL

Team member:

jd4573, hx2163

Description of our Application:

We designed an apartment management database system. The main two entries are Apartments and Tenants. These two entries are connected by Contracts entry. The main usage of this application is to help Apartment companies to manage the tenant information and check whether tenants have made the monthly payment on time. Moreover, the employee at the front desk can check whether the car parked in front of the front door is the tenant's, to make sure they don't pull the tenant's car.

Entity Sets:

Buildings, Apartments, Employees, Tenants, Contracts, Payments, Cars, Late_Fees

Relationship Set:

Business Rules:

Building:
Each building must have some apartment units.

Apartment:
Each apartment belongs to one building.

Employees:
Each employee works at some building.

Contracts:
Each contract is created by exactly one employee,
Each employee can create 0,1 or multiple contracts.

Tenants:
Each contract is signed by exactly one tenant.
Each tenant must be signed on some contracts.

Cars:
Some tenants have cars, this is a weak entity.

Payments:
Each contract has multiple payments.

LateFees:
This is a weak entity to payments, it may or may not have late fees.

Relational Schema:

```
drop table if exists Apartments cascade;
drop table if exists Buildings cascade;
drop table if exists Contains cascade;
drop table if exists Employees cascade;
drop table if exists Contracts cascade;
drop table if exists Created cascade;
drop table if exists Tenants cascade;
drop table if exists Cars cascade;
drop table if exists Has cascade;
drop table if exists Payments cascade;
drop table if exists Late_Fees cascade;
drop table if exists Paid cascade;
drop table if exists has_late_fee;



create table Apartments (
   apartment_id        integer primary key,
   number_of_bedrooms  integer,
   sqrt_feet           float,
   price               float
);

create table Buildings(
   building_id             integer primary key,
   building_name           varchar(128),
   number_of_apartments    integer
);

create table Contains(
   apartment_id        integer,
   building_id         integer,
   primary key (building_id,apartment_id),
   foreign key (apartment_id) references Apartments(apartment_id),
```

```sql
   foreign key (building_id) references Buildings(building_id)
);

create table Employees (
   employee_id          integer primary key,
   first_name           varchar(128),
   last_name            varchar(128),
   gender               varchar(1)
);

create table Contracts (
   contract_id          integer primary key,
   employee_id          integer,
   SSN                  integer,
   apartment_id         integer,
   start_date           date,
   end_date             date,
   deposit              float
);

create table Created (
   contract_id          integer,
   employee_id          integer,
   primary key (contract_id,employee_id),
   foreign key (contract_id) references Contracts(contract_id),
   foreign key (employee_id) references Employees(employee_id)
);

create table Tenants (
   SSN                  integer primary key,
   first_name           varchar(128),
   last_name            varchar(128),
   gender               varchar(1),
   contact_number       integer,
   email                varchar(128)
);

create table Cars (
   plate_number    varchar(128) primary key,
   model           varchar(128),
   make            varchar(128)
);
```

```sql
create table Has (
   SSN                 integer,
   plate_number        varchar(128),
   primary key(SSN,plate_number),
   foreign key (SSN) references Tenants(SSN) on delete cascade,
   foreign key (plate_number) references Cars(plate_number) on delete cascade
);

create table Payments(
   payment_id      integer primary key,
   rent_id             integer,
   payment_amount      float,
   payment_date        date
);

create table Late_Fees(
   late_id             integer primary key,
   late_fee            float
);

create table Paid(
   contract_id         integer,
   payment_id          integer,
   rent_fee            float,
   primary key (contract_id,payment_id),
   foreign key (contract_id) references Contracts(contract_id),
   foreign key (payment_id) references Payments(payment_id)
);

create table has_late_fee(
   late_id             integer,
   payment_id          integer,
   primary key(late_id,payment_id),
   foreign key (late_id) references Late_Fee(late_id) on delete cascade,
   foreign key (payment_id) references Payments(payment_id) on delete cascade
);
```