FINAL PROJECT: Apartment Management System

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:{ building_id, building_name, number_of_apartments }
Apartments: {apartment_id,number_of_bedrooms,sqrt_feet,price,building_id}
Employees: {employee_id,first_name,last_name,gender,work_from,work_to}
Tenants: {SSN,first_name,last_name,gender,contact_number,email}
Contracts: {contract_id,employee_id,tenant_ssn,apartment_id,start_date,end_date,deposit}
Payments: {payment_id,contract_id,payment_amount,payment_date}
Cars: {SSN,plate_number,model,make}
Late_Fees: {late_id,late_fee,payment_id}

Relationship Set:

Works_at: {building_id,employee_id}
Paid: {contract_id,payment_id}

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:

```sql
drop table if exists Buildings cascade;
drop table if exists Apartments cascade;
drop table if exists Employees cascade;
drop table if exists works_at cascade;
drop table if exists Tenants cascade;
drop table if exists Contracts cascade;
drop table if exists Cars cascade;
drop table if exists Payments cascade;
drop table if exists Paid cascade;
drop table if exists Late_Fees cascade;


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

create table Apartments(
    apartment_id        integer primary key,
    number_of_bedrooms  integer,
    sqrt_feet           float,
    price               float,
    building_id         integer not null,
    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),
    work_from           date,
    work_to             date
);
```

```sql
create table works_at(
    building_id         integer,
    employee_id         integer,
    primary key(building_id, employee_id),
    foreign key(employee_id) references Employees(employee_id),
    foreign key(building_id) references Buildings(building_id)
);

create table Tenants (
    SSN                 varchar(11) primary key,
    first_name          varchar(128),
    last_name           varchar(128),
    gender              varchar(1),
    contact_number      varchar(10),
    email               varchar(128)
);

create table Contracts (
    contract_id         integer primary key,
    employee_id         integer not null,
    tenant_SSN          varchar(11) not null unique,
    apartment_id        integer not null unique,
    start_date          date not null,
    end_date            date not null,
    deposit             float,
    foreign key (employee_id) references Employees(employee_id),
    foreign key (tenant_SSN) references Tenants(SSN)
);


create table Cars(
    SSN                 varchar(11) not null,
    plate_number        varchar(128),
    model               varchar(128),
    make                varchar(128),
    primary key(SSN,plate_number),
    foreign key (SSN) references Tenants(SSN) on delete cascade
);
```

```sql
create table Payments(
    payment_id          integer primary key,
    contract_id         integer,
    payment_amount      float,
    payment_date        date,
    foreign key (contract_id) references Contracts(contract_id)
);

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

create table Late_Fees(
    late_id             integer,
    late_fee            float,
    payment_id          integer not null,
    primary key(late_id,payment_id),
    foreign key (payment_id) references Payments(payment_id) on delete
cascade
);
```

SQL query :

Q1: Find tenants who is paying late

Input: tenants's SSN

```sql
select T.first_name, T.last_name, T.contact_number
from Tenants T, Payments P, Late_Fees L, Contracts C
where T.ssn = C.tenant_SSN
AND P.contract_id = C.contract_id
AND P.payment_id = L.payment_id
AND payment_date > 'input';
```

Q2: Find tenants name and contact number who live in the specific apartment,

Sort results by tenant's first name, with ties broken by tenant's last name.

Input: apartment number

```sql
select T.first_name, T.last_name, T.contact_number, A.apartment_id
from Tenants T, Apartments A, Contracts C
where T.ssn = C.tenant_SSN
AND C.apartment_id = A.apartment_id
AND A.apartment_id = 'apartment number'
ORDER BY T.first_name, T.last_name;
```

Q3: Find all the female employees who are on active duty and work more than 1 year Sort results by employees first name, with ties broken by employees last name.

Input: 365

```sql
Select CurrentEmployees.first_name, CurrentEmployees.last_name,
CurrentEmployees.workingDay
FROM(
    select E.first_name, E.last_name, TRUNC(DATE_PART('day',
CURRENT_DATE::timestamp - E.work_from::timestamp)) AS workingDay
    FROM Employees E
    WHERE E.work_to IS NOT NULL
    AND E.gender = 'F'
) AS CurrentEmployees
WHERE CurrentEmployees.workingDay >= 365
ORDER BY CurrentEmployees.first_name, CurrentEmployees.last_name;
```

Q4: Find number of tenants who live in the same building, Sort results by building name.

Input: Yes

```sql
select B.building_id ,count(B.building_id) AS Num_people
from Contracts C, Buildings B, Apartments A
WHERE C.apartment_id = A.apartment_id
AND A.building_id = B.building_id
Group BY B.building_id
Order BY B.building_name;
```

Q5: Find total payment and fees for specific tenants.

Input: tenant's SSN

```sql
SELECT T.first_name, T.last_name, SUM(P.payment_amount) AS total_payment,
Fee_table.total_fee
FROM Payments P, Tenants T, Contracts C,
(
    SELECT P.contract_id, SUM(L.late_fee) AS total_fee
    FROM Late_Fees L, Paid P
    WHERE L.payment_id = P.payment_id
    GROUP BY contract_id
) AS Fee_table
WHERE T.SSN = C.tenant_ssn
AND P.contract_id = C.contract_id
AND Fee_table.contract_id = P.contract_id
AND T.SSN = 'SSN'
GROUP BY T.SSN, Fee_table.total_fee;
```

Q6: Find all tenants who have cars and live in building 2, Sort results by tenant's first name, with ties broken by tenant's last name and then by plate number.

Input: building_id

```sql
SELECT T.first_name, T.last_name, Cars.plate_number
FROM Tenants T, Cars, Contracts C,
(
    Select C.contract_id
    From Contracts C, Buildings B, Apartments A
    WHERE C.apartment_id = A.apartment_id
    AND A.building_id = B.building_id
    AND B.building_id = '2'
) AS B2Contracts
WHERE T.SSN = C.tenant_ssn
AND C.contract_id = B2Contracts.contract_id
AND T.SSN = Cars.SSN
ORDER BY T.first_name, T.last_name, Cars.plate_number;
```

Q7: Find number of empty apartments in each building, Sort results by building name.

Input: Yes

```sql
SELECT B.building_name, (B.number_of_apartments -
ApartmentsLeft.rented_room) AS empty_apartments
FROM Buildings B,
(
    Select A.building_id, count(*) AS rented_room
    From Contracts C, Apartments A
    WHERE C.apartment_id = A.apartment_id
    AND C.end_date > CURRENT_DATE
    GROUP BY A.building_id
) AS ApartmentsLeft
WHERE B.building_id = ApartmentsLeft.building_id
ORDER BY B.building_name;
```

Q8: List pairs of tenants from the Tenants table s.t. One of them had cars and the other did not. Sort results by Tenant_has_car, with ties broken by Tenant_has_no_car.

Input: Yes

```sql
Select (T1.first_name , T1.last_name) as Tenant_has_car, (T2.first_name ,
T2.last_name) as Tenant_has_no_car
FROM Tenants T1, Tenants T2, Cars C,
(
    Select (T.first_name, T.last_name) as Nocar_tenant,T.SSN
    FROM Tenants T
    EXCEPT
    SELECT (T.first_name, T.last_name) as Has_car, T.SSN
    FROM Tenants T, Cars C
    WHERE T.SSN = C.SSN
) AS NoCar
WHERE T1.SSN = C.SSN
AND T2.SSN = NoCar.SSN
GROUP BY Tenant_has_car,Tenant_has_no_car
ORDER BY Tenant_has_car,Tenant_has_no_car;
```