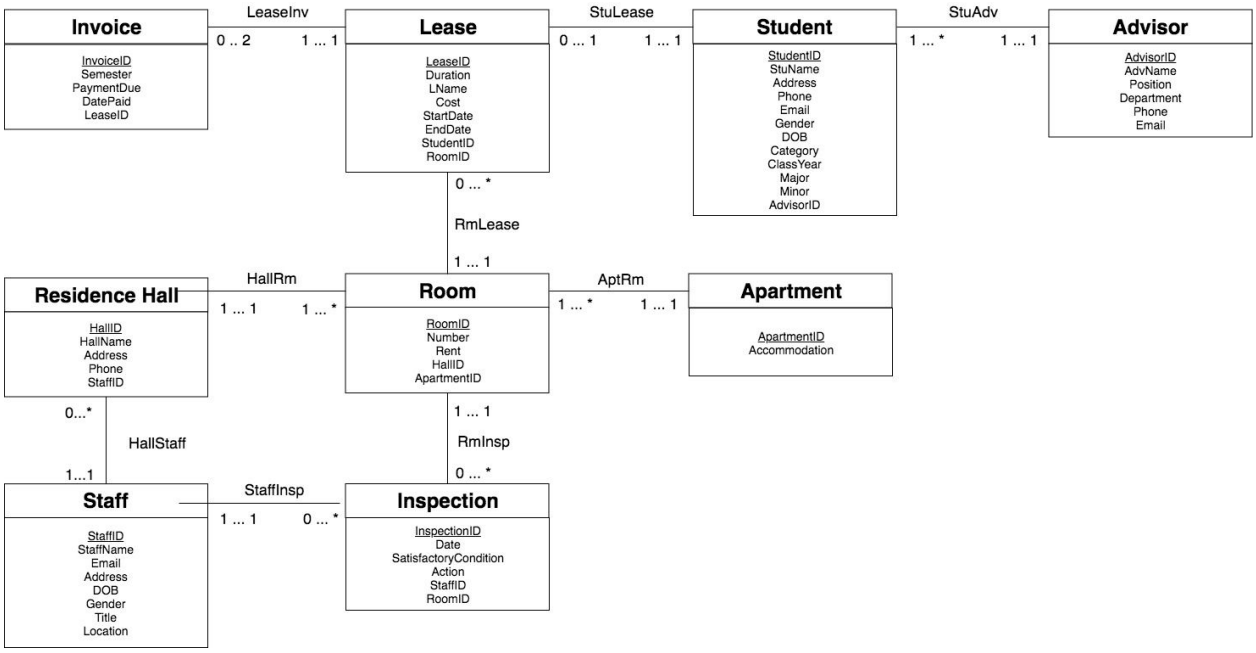
I. Conceptual Database Design:



Rationale: One piece that may stand out in this database design UML is that we decided to make a central table for all rooms on campus. Meaning a room from an residence hall was no different than a room from an apartment. We assumed that a student would always make a lease on an individual room and if a student wanted to, they could lease out an entire apartment if they just make multiple leases for each room. All tables have a single primary key with all of them being artificially made. The way someone could distinguish if a room came from an apartment or from a residence hall is if the foreign key HallID or ApartmentID were NULL in the database.

II. Logical Database Design:

Invoice

InvoiceID	Semester	PaymentDue	DatePaid	LeaseID

Lease

LeaseID	Duration	LName	Cost	StartDate	EndDate	StudentID	RoomID

Student

StudentID	StuName	Address	Phone	Email	Gender	DOB	Category	ClassYear	Major	Minor	AdvisorID

Advisor

AdvisorID	AdvName	Position	Department	Phone	Email

Residence_Hall

HallID	HallName	Address	Phone	StaffID

Room

RoomID	RoomNum	Rent	HallID	ApartmentID

Apartment

ApartmentID	Accomodation

Staff

StaffID	StaffName	Email	Address	DOB	Gender	Title	Location

Inspection

InspectionID	DateInspected	SatisfactoryCondition	Action	StaffID	RoomID

Above are the actual table headings that we used. Shown are the primary keys that we used for each table. All foreign have the same names as the primary keys of the matching table. All primary keys are made up of a single key that has been artificially created. For most of the tables this was done with a counter. Some constraints were implemented on the tables when they were created according to what the spec sheet wanted. The code for the tables created is shown below.

```
CREATE TABLE staff (
   StaffID int NOT NULL,
   StaffName varchar2(100) NOT NULL,
   Email varchar2(100),
   Address varchar2(100) NOT NULL,
   DOB varchar2(20) NOT NULL,
   Gender varchar2(1) NOT NULL CHECK (Gender IN ('M', 'F')),
   Title varchar2(50) NOT NULL CHECK (Title IN ('Hall Manager', 'Cleaner', 'Administrative
Assistant')),
   Location varchar2(50) NOT NULL CHECK (Location IN ('Residence Office', 'Residence
Hall')),
   PRIMARY KEY (StaffID)
);
```

```
CREATE TABLE residence_hall (
   HallID varchar2(10) NOT NULL,
   HallName varchar2(100) NOT NULL,
   Address varchar2(100) NOT NULL,
   Phone varchar2(15) NOT NULL,
   StaffID int,
   PRIMARY KEY (HallID),
   FOREIGN KEY (StaffID) REFERENCES staff (StaffID)
);

CREATE TABLE apartment (
   ApartmentID varchar2(10) NOT NULL,
   Accomodation int NOT NULL,
   PRIMARY KEY (ApartmentID)
);

CREATE TABLE room (
   RoomID int NOT NULL,
   RoomNum int NOT NULL,
   Rent float NOT NULL,
   HallID varchar2(10),
   ApartmentID varchar2(20),
   PRIMARY KEY (RoomID),
   FOREIGN KEY (HallID) REFERENCES residence_hall (HallID),
   FOREIGN KEY (ApartmentID) REFERENCES apartment (ApartmentID)
);

CREATE TABLE inspection (
   InspectionID int NOT NULL,
   DateInspected varchar2(20) NOT NULL,
   SatisfactoryCondition varchar2(1) CHECK (SatisfactoryCondition IN ('Y', 'N')),
   Action varchar2(300),
   StaffID int,
   RoomID int NOT NULL,
   PRIMARY KEY (InspectionID),
   FOREIGN KEY (StaffID) REFERENCES staff (StaffID),
   FOREIGN KEY (RoomID) REFERENCES room (RoomID)
);

CREATE TABLE advisor (
   AdvisorID int NOT NULL,
   AdvName varchar2(100) NOT NULL,
   Position varchar2(100) NOT NULL CHECK (Position IN ('Assistant', 'Head')),
```

```
    Department varchar2(50) NOT NULL,
    Phone varchar2(15),
    Email varchar2(300),
    PRIMARY KEY (AdvisorID)
);

CREATE TABLE student (
    StudentID int NOT NULL,
    StuName varchar2(100) NOT NULL,
    Address varchar2(100) NOT NULL,
    Phone varchar2(15) NOT NULL,
    Email varchar2(300),
    Gender varchar2(1) NOT NULL CHECK (Gender IN ('M', 'F')),
    DOB varchar2(20) NOT NULL,
    Category varchar2(14) NOT NULL CHECK (Category IN ('undergraduate', 'postgraduate')),
    ClassYear varchar2(15) CHECK (ClassYear IN ('first-year', 'second-year', 'third-year',
'fourth-year')),
    Major varchar2(50),
    Minor varchar2(50),
    AdvisorID int NOT NULL,
    PRIMARY KEY (StudentID),
    FOREIGN KEY (AdvisorID) REFERENCES advisor (AdvisorID)
);

CREATE TABLE lease (
    LeaseID int NOT NULL,
    Duration int NOT NULL CHECK (Duration=1 OR Duration=2),
    LName varchar2(100) NOT NULL,
    Cost float NOT NULL,
    StartDate date NOT NULL,
    EndDate date NOT NULL,
    StudentID int NOT NULL,
    RoomID int NOT NULL,
    CONSTRAINT CheckEndLaterThanStart CHECK (EndDate >= StartDate),
    PRIMARY KEY (LeaseID),
    FOREIGN KEY (StudentID) REFERENCES student (StudentID),
    FOREIGN KEY (RoomID) REFERENCES room (RoomID)
);

CREATE TABLE invoice (
    InvoiceID int NOT NULL,
    Semester varchar2(10) NOT NULL CHECK (Semester IN ('Fall', 'Spring')),
    PaymentDue float NOT NULL,
```

```
    DatePaid date,
    LeaseID int NOT NULL,
    PRIMARY KEY (InvoiceID),
    FOREIGN KEY (LeaseID) REFERENCES lease (LeaseID)
);
```

III. Normalization Analysis:

FDs:

- InvoiceID → PaymentDue
- LeaseID → StartDate
  LeaseID → EndDate
  LeaseID → Duration
- StudentID → Name
  StudentID → Advisor
  StudentID → Category
  StudentID → ClassYear
- AdvisorID → Name
  AdvisorID → Department
- HallID → Name
  HallID → Address
  HallID → StaffID
- RoomID → RoomNum
  RoomID → Rent
- ApartmentID → Accommodation
- StaffID → Name
- InspectionID → Date
  InspectionID → Status
  InspectionID → Action

Analysis: By definition of third normal form a relation R which would be all of the tables above, is in 3NF if, for every non-trivial FD X → A (All FDs above are non-trivial) that holds in the relations, either X is a super key or A is a prime attribute. All of the left hand sides of the FDs are all superkeys because they are the only key used in each table, thus all the relationships are in 3NF.

IV. Query Description:

Query One: This query answered the question of which rooms were currently vacant at the university. Although the rooms might have a lease in the future this query told the user what rooms did not have any active leases. This is a useful query when making a lease so a user does not create an active lease on a room that already has a person living in the room.

Query Two: This query did something with the inspections table. Specifically it took in some user input. The user could select a drop down menu of all the rooms in the university database and allowed the user to get a list of inspection information on a specific room. Information included was what staff member did the inspection, the date of inspection and what actions were taken if the inspection was not satisfactory. This information would be useful in a system like this in order to add up expenses of all of the fixes that were made to the rooms.