University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: October 15, 2024

Group Number: 48

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Jan Smailbegovic	59752725	y8z7i	Jan.smailbegovic@gmail.com
Aayush Behl	26071365	p7h3b	aayush.behl32@gmail.com
Alex Briauzov	85706752	b4u3g	abryauzo@student.ubc.ca

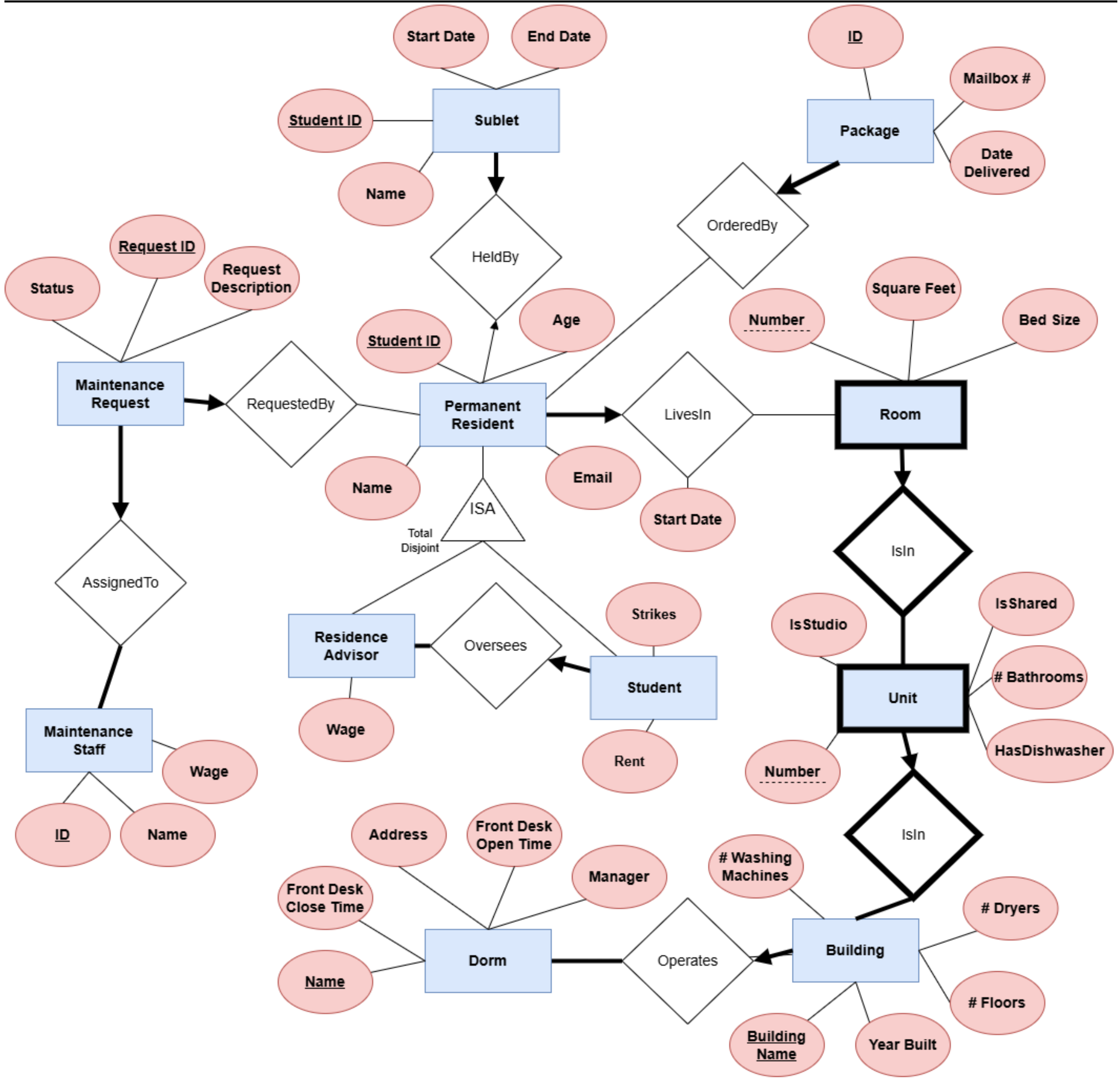
University of British Columbia, Vancouver
Department of Computer Science

Summary:
The goal of our project is to represent the current state of a university residence. To do this, we will store information about the state of each dorm, as well as who is currently living in each room. Our database will also keep track of resident activity by storing maintenance requests, sublets and package deliveries.

ER Diagram:
Changes:

- Fixed the participation constraint in the HeldBy relationship between PermanentResident and Sublet to make holding sublets *optional* for PermanentResident
- Updated the partial keys in the weak entities, Room and Unit, to be represented with a dotted line instead of a solid line

Note: we will keep the primary key in Sublet as StudentID (against the suggestion of our mentor) because the purpose of our database is to model the current state of the residence, meaning each permanent resident will only ever have, at most, one sublet. And, because a subleasee cannot stay in two sublets at the same time, StudentID is a valid and sufficient primary key.


Start Date
End Date
ID
Mailbox #
Student ID
Sublet
Package
Date Delivered
Name
OrderedBy
HeldBy
Request ID
Status
Request Description
Student ID
Age
Number
Square Feet
Bed Size
Maintenance Request
RequestedBy
Permanent Resident
LivesIn
Room
Name
Email
Start Date
ISA
Total Disjoint
IsIn
AssignedTo
Strikes
IsStudio
IsShared
Residence Advisor
Oversees
Student
Unit
# Bathrooms
Wage
HasDishwasher
Maintenance Staff
Rent
Number
Wage
IsIn
ID
Name
Address
Front Desk Open Time
# Washing Machines
Manager
Front Desk Close Time
# Dryers
Dorm
Operates
Building
Name
# Floors
Building Name
Year Built

Derived Schemas:
Dorm(name: char[50], address: char[50], frontDeskOpenTime: char[10], frontDeskCloseTime: char[10], manager: char[50])
CKs: name, address
PK: name
Constraints:
- name: unique, not null
- address: unique, not null
- manager: not null

Building(buildingName: char[50], dormName: char[50], yearBuilt: integer, floors: integer, washingMachines: integer, dryers: integer)
CKs: buildingName
FKs: dormName
PK: buildingName
Constraints:
- buildingName: unique, not null
- dormName: not null, references a Dorm
- yearBuilt: not null
- floors: not null

Unit(number: integer, buildingName: char[50] isStudio: boolean, isShared: boolean, bathrooms: integer, hasDishwasher: boolean)
CKs: {number, buildingName}
FKs: buildingName
PK: {number, buildingName}
Constraints:
- number: not null
- buildingName: not null, references a Building
- isStudio: not null

Room(number: integer, unitNumber: integer, buildingName: char[50], sqFeet: integer, bedSize: char[20])
CKs: {number, unitNumber, buildingName}
FKs: unitNumber, buildingName
PK: {number, unitNumber, buildingName}
Constraints:
- number: not null
- unitNumber: not null, references a Unit
- buildingName: not null, references a Building
- sqFeet: not null

PermanentResident(studentId: integer, roomNumber: integer, unitNumber: integer, buildingName: char[50], subletId: integer, age: integer, name: char[50], email: char[50])
CKs: studentId, email
FKs: roomNumber, unitNumber, buildingName, subletId
PK: studentId
Constraints:

- studentId: unique, not null
- roomNumber: not null, references a Room
- unitNumber: not null, references a Room,
- buildingName: not null, references a Room,
- subletId: references a Sublet
- name: not null
- email: unique, not null

ResidenceAdvisor(studentId: integer, biweeklyWage: integer)
CKs: studentId
FKs: studentId
PK: studentId
Constraints:

- studentId: unique, not null, references a PermanentResident
- biweeklyWage: not null

Student(studentId: integer, residenceAdvisorId: integer, strikes: integer, rent: integer)
CKs: studentId
FKs: studentId, residenceAdvisorId
PK: studentId
Constraints

- studentId: unique, not null, references a PermanentResident
- residenceAdvisorId: not null, references a ResidenceAdvisor
- strikes: not null
- rent: not null

MaintenanceRequest(requestId: integer, studentId: integer, staffId: integer, description: char[1000], status: char[20])
CKs: requestId
FKs: studentId, staffId
PK: requestId
Constraints:

- requestId: unique, not null
- studentId: not null, references a PermanentResident
- staffId: references a MaintenanceStaff
- description: not null
- status: not null

MaintenanceStaff(staffId: integer, name: char[50], biweeklyWage: integer)
CKs: staffId
PK: staffId
Constraints:

- staffId: unique, not null
- name: not null
- biweeklyWage: not null

Sublet(studentId: integer, residentId: integer, name: char[50], startDate: char[50], endDate: char[50])
CKs: studentId, residentId
FKs: residentId
PK: studentId
Constraints:

- studentId: unique, not null
- residentId: unique, not null, references a PermanentResident
- name: not null
- startDate: not null
- endDate: not null

Package(id: integer, studentId: integer, mailbox: integer, deliveryDate: char[50])
CKs: id
FKs: studentId
PK: id
Constraints:

- id: unique, not null
- studentId: not null, references a PermanentResident
- mailbox: not null

Functional Dependencies:

Dorm (name, address, frontDeskOpenTime, frontDeskCloseTime, manager)
name -> {address, frontDeskOpenTime, frontDeskCloseTime, manager}
address -> {name, frontDeskOpenTime, frontDeskCloseTime, manager}

Building (buildingName, dormName, yearBuilt, floors, washingMachines, dryers)
buildingName -> {dormName, yearBuilt, floors, washingMachines, dryers}
floors -> {washingMachines, dryers}

Unit (number, buildingName, isStudio, isShared, bathrooms, hasDishwasher)
{number, buildingName} - > {isStudio, isShared, bathrooms, hasDishwasher}

Room (number, unitNumber, buildingName, sqFeet, bedSize)
{number, unitNumber, buildingName} -> {sqFeet, bedSize}
sqFeet -> bedSize

PermanentResident (studentId, roomNumber, unitNumber, buildingName, subletId, age, name, email)
studentId -> {roomNumber, unitNumber, buildingName, subletId, age, name, email}
email -> {roomNumber, unitNumber, buildingName, subletId, age, name, studentId}

ResidenceAdvisor (studentId, biweeklyWage)
studentId -> biweeklyWage

Student (studentId, residenceAdvisorId, strikes, rent)
studentId - > {residenceAdvisorId, strikes, rent}

MaintenanceRequest (requestId, studentId, staffId, description, status)
requestId -> {studentId, staffId, description, description, status}

MaintenanceStaff (staffId, name, biweeklyWage)
staffId -> {name, biweeklyWage}

Sublet (studentId, residentId, name, startDate, endDate)
studentId -> {residentId, name, startDate, endDate}
residentId -> {studentId, name, startDate, endDate}

Package (id, studentId, mailbox, deliveryDate)
Id -> {studentId, mailbox, deliveryDate}

Normalization:

Dorm (name, address, frontDeskOpenTime, frontDeskCloseTime, manager)
FDs:
name -> address, frontDeskOpenTime, frontDeskCloseTime, manager
address - > name, frontDeskOpenTime, frontDeskCloseTime, manager

Keys:
name+: {name, address, frontDeskOpenTime, frontDeskCloseTime, manager}
address+: {name, address, frontDeskOpenTime, frontDeskCloseTime, manager}

No functional dependencies violate BCNF, as each LHS is a super key of the relation.

R1 and R2 contain the same attributes, so we can omit R2.

Decomposition in BCNF:
R1(<u>name</u>, address, frontDeskOpenTime, frontDeskCloseTime, manager)

Building (buildingName, dormName, yearBuilt, floors, washingMachines, dryers)
FDs:
buildingName -> {dormName, yearBuilt, floors, washingMachines, dryers}
floors -> {washingMachines, dryers}

Keys:
buildingName+: {buildingName, dormName, yearBuilt, floors, washingMachines, dryers}

floors -> {washingMachines, dryers} violates BCNF.

Decomposition in BCNF:
R1(<u>buildingName</u>, dormName, yearBuilt, floors)
R2(<u>floors</u>, washingMachines, dryers)

Unit (number, buildingName, isStudio, isShared, bathrooms, hasDishwasher)
FDs:
{number, buildingName} - > {isStudio, isShared, bathrooms, hasDishwasher}

Keys:
{number, buildingName}+: {number, buildingName, isStudio, isShared, bathrooms, hasDishwasher}

No functional dependencies violate BCNF, as each LHS is a super key of the relation.

Decompostion into BCNF:
R1(number, buildingName, isStudio, isShared, bathrooms, hasDishwasher)

Room (number, unitNumber, buildingName, sqFeet, bedSize)
FDs:
{number, unitNumber, buildingName} -> {sqFeet, bedSize}
sqFeet -> bedSize

Keys:
{number, unitNumber, buildingName}+: {number, unitNumber, buildingName, sqFeet, bedSize}

sqFeet -> bedSize violates BCNF

Decomposition into BCNF:
R1(sqFeet, bedSize)
R2(number, unitNumber, buildingName, sqFeet)

PermanentResident (studentId, roomNumber, unitNumber, buildingName, subletId, age, name, email)
FDs:
studentId -> {roomNumber, unitNumber, buildingName, subletId, age, name, email}
email -> {roomNumber, unitNumber, buildingName, subletId, age, name, studentId}

Keys:
studentId+: {studentId, roomNumber, unitNumber, buildingName, subletId, age, name, email}
email+: {studentId, roomNumber, unitNumber, buildingName, subletId, age, name, email}

No functional dependencies violate BCNF, as each LHS is a super key of the relation.

R1 and R2 contain the same attributes, so we can omit R2.

Decomposition into BCNF:
R1(<u>studentId</u>, roomNumber, unitNumber, buildingName, subletId, age, name, email)

ResidenceAdvisor (studentId, biweeklyWage)

FDs:
studentId -> biweeklyWage

Keys:
StudentId+: {studentId, biweeklyWage}

No functional dependencies violate BCNF, as each LHS is a super key of the relation.

Decomposition into BCNF:
R1(<u>studentId</u>, biweeklyWage)

Student (studentId, residenceAdvisorId, strikes, rent)
FDs:
studentId -> {residenceAdvisorId, strikes, rent}

Keys:
studentId+: {studentId, residenceAdvisorId, strikes, rent}

No functional dependencies violate BCNF as each LHS is a super key of the relation.

Decomposition into BCNF:
R1(<u>studentId</u>, residenceAdvisorId, strikes, rent)

MaintenanceRequest (requestId, studentId, staffId, description, status)
FDs:
requestId -> {studentId, staffId, description, description, status}

Keys:
requestId+: {requestId, studentId, staffId, description, description, status}

No functional dependencies violate BCNF as each LHS is a super key of the relation.

Decomposition into BCNF:
R1(<u>requestId</u>, studentId, staffId, description, description, status)

MaintenanceStaff (staffId, name, biweeklyWage)
FDs:
staffId -> {name, biweeklyWage}

Keys:
staffId+:{staffId, name, biweeklyWage}

No functional dependencies violate BCNF, as each LHS is a super key of the relation.

Decomposition into BCNF:
R1(<u>staffId</u>, name, biweeklyWage)

Sublet (studentId, residentId, name, startDate, endDate)
FDs:
studentId -> {residentId, name, startDate, endDate}
residentId -> {studentId, name, startDate, endDate}

Keys:
studentId+: {studentId, residentId, name, startDate, endDate}
residentId+: {studentId, residentId, name, startDate, endDate}

No functional dependencies violate BCNF, as each LHS is a super key of the relation.

R1 and R2 contain the same attributes, so we can omit R2.

Decomposition into BCNF:
R1(studentId, residentId, name, startDate, endDate)

Package (id, studentId, mailbox, deliveryDate)
FDs:
Id -> {studentId, mailbox, deliveryDate}

Keys:
Id+: {Id, studentId, mailbox, deliveryDate}

No functional dependencies violate BCNF, as each LHS is a super key of the relation.

Decomposition into BCNF:
R1(Id, studentId, mailbox, deliveryDate)

SQL DDL Statements:

```
CREATE TABLE Dorm (
	name VARCHAR(50) UNIQUE NOT NULL,
	address VARCHAR(50) UNIQUE NOT NULL,
	frontDeskOpenTime VARCHAR(10),
	frontDeskCloseTime VARCHAR(10),
	manager VARCHAR(50) NOT NULL,
	PRIMARY KEY (name)
)

CREATE TABLE Building_R2 (
	floors INTEGER UNIQUE NOT NULL,
	washingMachines  INTEGER,
	dryers INTEGER,
	PRIMARY KEY (floors)
)

CREATE TABLE Building_R1 (
	buildingName VARCHAR(50) UNIQUE NOT NULL,
	dormName  VARCHAR(50) NOT NULL,
	yearBuilt INTEGER NOT NULL,
	floors INTEGER NOT NULL,
	PRIMARY KEY (buildingName),
	FOREIGN KEY (dormName) REFERENCES Dorm(name),
	FOREIGN KEY (floors) REFERENCES Building_R2(floors)
		ON DELETE CASCADE
)

CREATE TABLE Unit (
	number INTEGER NOT NULL,
	buildingName VARCHAR(50) NOT NULL,
	isStudio BOOLEAN NOT NULL,
	isShared BOOLEAN,
	bathrooms INTEGER,
	hasDishwasher BOOLEAN,
	PRIMARY KEY (number, buildingName),
	FOREIGN KEY (buildingName) REFERENCES Building_R1(buildingName)
)
```

```
CREATE TABLE Room_R1(
        sqFeet INTEGER UNIQUE NOT NULL,
        bedSize VARCHAR,
        PRIMARY KEY (sqFeet)
)

CREATE TABLE Room_R2(
        number INTEGER NOT NULL,
        unitNumber INTEGER NOT NULL,
        buildingName VARCHAR(50) NOT NULL,
        sqFeet INTEGER NOT NULL,
        PRIMARY KEY (number, unitNumber, buildingName),
        FOREIGN KEY (sqFeet) REFERENCES Room_R1(sqFeet)
                ON DELETE CASCADE,
        FOREIGN KEY (unitNumber, buildingName)
                REFERENCES Unit(number, buidingName)
)

CREATE TABLE PermanentResident (
        studentId INTEGER UNIQUE NOT NULL,
        roomNumber INTEGER NOT NULL,
        unitNumber INTEGER NOT NULL,
        buildingName VARCHAR(50) NOT NULL,
        subletId INTEGER
        age  INTEGER,
        name VARCHAR(50) NOT NULL,
        email VARCHAR(50) UNIQUE NOT NULL,
        PRIMARY KEY (studentId),
        FOREIGN KEY (roomNumber, unitNumber, buildingName)
                REFERENCES Room_R2 (roomNumber, unitNumber, buildingName),
        FOREIGN KEY (subletId) REFERENCES Student(subletId)
)

CREATE TABLE ResidentAdvisor (
        studentId INTEGER UNIQUE NOT NULL,
        biweeklyWage INTEGER NOT NULL,
        PRIMARY KEY (studentId),
        FOREIGN KEY (studentId) REFERENCES PermanentResident(studentId)
                ON DELETE CASCADE
)
```

```
CREATE TABLE Student (
        studentId INTEGER UNIQUE NOT NULL,
        residenceAdvisorId INTEGER NOT NULL,
        strikes INTEGER NOT NULL,
        rent INTEGER NOT NULL,
        PRIMARY KEY (studentId),
        FOREIGN KEY (studentId) REFERENCES PermanentResident(studentId)
                ON DELETE CASCADE,
        FOREIGN KEY (residenceAdvisorId) REFERENCES ResidenceAdvisor(studentId)
)

CREATE TABLE MaintenanceRequest(
        requestId INTEGER UNIQUE NOT NULL,
        studentId INTEGER NOT NULL,
        staffId INTEGER,
        description VARCHAR(1000) NOT NULL,
        status VARCHAR(20) NOT NULL,
        PRIMARY KEY (requestId),
        FOREIGN KEY (studentId) REFERENCES PermanentResident(studentId),
        FOREIGN KEY (staffId) REFERENCES MaintenanceStaff(staffId)
)

CREATE TABLE MaintenanceStaff (
        staffId INTEGER UNIQUE NOT NULL,
        name VARCHAR(50) NOT NULL,
        biweeklyWage INTEGER NOT NULL,
        PRIMARY KEY (staffId)
)

CREATE TABLE Sublet (
        studentId INTEGER UNIQUE NOT NULL,
        residentId INTEGER UNIQUE NOT NULL,
        name VARCHAR(50) NOT NULL,
        startDate VARCHAR(50) NOT NULL,
        endDate VARCHAR(50) NOT NULL,
        PRIMARY KEY (studentId),
        FOREIGN KEY (residentId) REFERENCES PermanentResident(studentId)
)
```

```
CREATE TABLE Package (
        id INTEGER NOT NULL,
        studentId INTEGER NOT NULL,
        mailbox INTEGER NOT NULL,
        deliveryDate VARCHAR(50),
        PRIMARY KEY (id),
        FOREIGN KEY (studentId) REFERENCES PermanentResident(studentId)
)
```

SQL INSERT Statements:

```
INSERT
INTO Dorm (name, address, frontDeskOpenTime, frontDeskCloseTime, manager)
VALUES ("Marine Drive”, "2205 Lower Mall, Vancouver, BC V6T 1Z4”, “9:00”,
“22:00”, "Malak Aiad”)
INSERT
INTO Dorm (name, address, frontDeskOpenTime, frontDeskCloseTime, manager)
VALUES ("Exchange”, "5955 Student Union Blvd, Vancouver, BC V6T 1K2”, “9:00”,
“22:00”, "Jennifer Gunn”)
INSERT
INTO Dorm (name, address, frontDeskOpenTime, frontDeskCloseTime, manager)
VALUES ("Ponderosa Commons”, "2075 West Mall, Vancouver, BC V6T 1Z2”, “9:00”,
“22:00”, "Jane Doe”)
INSERT
INTO Dorm (name, address, frontDeskOpenTime, frontDeskCloseTime, manager)
VALUES ("Orchard Commons”, "6363 Agronomy Road, Vancouver, BC V6T 1Z4”, “9:00”,
“22:00”, "Calvin Cheung”)
INSERT
INTO Dorm (name, address, frontDeskOpenTime, frontDeskCloseTime, manager)
VALUES ("Walter Gage”, "5959 Student Union Blvd. Vancouver, BC V6T 1K2”, “9:00”,
“22:00”, "Eniola Folarin”)

INSERT INTO Building_R2 (floors, washingMachines, dryers) VALUES (4, 4, 4)
INSERT INTO Building_R2 (floors, washingMachines, dryers) VALUES (6, 6, 6)
INSERT INTO Building_R2 (floors, washingMachines, dryers) VALUES (24, 10, 10)
INSERT INTO Building_R2 (floors, washingMachines, dryers) VALUES (10, 8, 8)
INSERT INTO Building_R2 (floors, washingMachines, dryers) VALUES (17, 8, 8)
INSERT INTO Building_R1 (buildingName, dormName, yearBuilt, floors)
VALUES (“Building 1”, “Marine Drive”, 2000, 17)
INSERT INTO Building_R1 (buildingName, dormName, yearBuilt, floors)
VALUES (“Building 2”, “Marine Drive”, 2007, 10)
INSERT INTO Building_R1 (buildingName, dormName, yearBuilt, floors)
VALUES (“Oak”, “Ponderosa Commons”, 2000, 24)
INSERT INTO Building_R1 (buildingName, dormName, yearBuilt, floors)
VALUES (“Cedar”, “Ponderosa Commons”, 2001, 24)
INSERT INTO Building_R1 (buildingName, dormName, yearBuilt, floors)
VALUES (“Braeburn”, “Orchard Commons”, 2016, 24)

INSERT INTO Unit (number, buildingName, isStudio, isShared, bathrooms, hasDishwasher)
VALUES (1, “Building 1”, true, false, 1, false)
INSERT INTO Unit (number, buildingName, isStudio, isShared, bathrooms, hasDishwasher)
VALUES (2, “Building 1”, false, true, 2, false)
INSERT INTO Unit (number, buildingName, isStudio, isShared, bathrooms, hasDishwasher)
VALUES (6, “Building 2”, false, true, 2, false)
INSERT INTO Unit (number, buildingName, isStudio, isShared, bathrooms, hasDishwasher)
VALUES (7, “Building 2”, true, false, 1, false)
INSERT INTO Unit (number, buildingName, isStudio, isShared, bathrooms, hasDishwasher)
VALUES (11, “Building 3”, true, true, 1, true)

INSERT INTO Room_R1 (sqFeet, bedSize) VALUES (80, “Twin XL”)
INSERT INTO Room_R1 (sqFeet, bedSize) VALUES (100, “Twin XL”)
INSERT INTO Room_R1 (sqFeet, bedSize) VALUES (120, “Twin XL”)
INSERT INTO Room_R1 (sqFeet, bedSize) VALUES (200, “Queen”)
INSERT INTO Room_R1 (sqFeet, bedSize) VALUES (250, “Queen”)

INSERT INTO Room_R2(number, unitNumber, buildingName, sqFeet) VALUES (2, 402, “Building
1”, 100)
INSERT INTO Room_R2(number, unitNumber, buildingName, sqFeet) VALUES (1, 402, “Building
1”, 100)
INSERT INTO Room_R2(number, unitNumber, buildingName, sqFeet) VALUES (2, 1402, “Cedar”,
120)
INSERT INTO Room_R2(number, unitNumber, buildingName, sqFeet) VALUES (1, 1202,
“Braeburn”, 200)
INSERT INTO Room_R2(number, unitNumber, buildingName, sqFeet) VALUES (2, 402, “Building
2”, 250)
```

```
INSERT INTO PermanentResident (studentId, roomNumber, unitNumber, buildingName,
subletId, age, name, email) VALUES (13849148,1,204, ”Oak”, null, 20, ”William Nylander”,
”willy@gmail.com”)
INSERT INTO PermanentResident (studentId, roomNumber, unitNumber, buildingName,
subletId, age, name, email) VALUES (71849184, 2, 204, ”Oak”, null, 21, ”Morgan Reilly”,
“mo.reilly@gmail.com”)
INSERT INTO PermanentResident (studentId, roomNumber, unitNumber, buildingName,
subletId, age, name, email) VALUES (81941048, 3, 204, “Oak”, null, 20, “Jane Matthews”,
“jane.matthews@hotmail.com”)
INSERT INTO PermanentResident (studentId, roomNumber, unitNumber, buildingName,
subletId, age, name, email) VALUES (48105829, 1, 7012, “Braeburn”, null, 19, “Emily Marner”,
“emily.marner@gmail.com”)
INSERT INTO PermanentResident (studentId, roomNumber, unitNumber, buildingName,
subletId, age, name, email) VALUES (71193810, 1, 101, “Cedar”, 81879912, 22, “Anna Tanev”,
“anna.tanev@gmail.com”)
INSERT INTO PermanentResident (studentId, roomNumber, unitNumber, buildingName,
subletId, age, name, email) VALUES (82957194, 2, 101, “Cedar”, 82957275, 22, “John Tavares”,
“jt91@gmail.com”)
INSERT INTO PermanentResident (studentId, roomNumber, unitNumber, buildingName,
subletId, age, name, email) VALUES (12368295, 3, 101, “Cedar”, null, 20, “Calle Jarnkrok”,
“calle.j@gmail.com”)
INSERT INTO PermanentResident (studentId, roomNumber, unitNumber, buildingName,
subletId, age, name, email) VALUES (22385013, 4, 506, “Cedar”, null, 21, “Mae Dae”,
“mdae@gmail.com”)
INSERT INTO PermanentResident (studentId, roomNumber, unitNumber, buildingName,
subletId, age, name, email) VALUES (44185018, 6, N0503, “North Tower”, null, 22, “Ally Sing”,
“asing@gmail.com”)
INSERT INTO PermanentResident (studentId, roomNumber, unitNumber, buildingName,
subletId, age, name, email) VALUES (28492572, 2, N0503, “North Tower”, null, 19, “Sarah
Robertson”, “s.robo@gmail.com”)

INSERT INTO ResidentAdvisor (studentId, biweeklyWage) VALUES (13849148, 500)
INSERT INTO ResidentAdvisor (studentId, biweeklyWage) VALUES (71849184, 600)
INSERT INTO ResidentAdvisor (studentId, biweeklyWage) VALUES (81941048, 450)
INSERT INTO ResidentAdvisor (studentId, biweeklyWage) VALUES (48105829, 500)
INSERT INTO ResidentAdvisor (studentId, biweeklyWage) VALUES (71193810, 700)

INSERT INTO Student (studentId, residenceAdvisorId, strikes, rent) VALUES (82957194,
13849148, 0, 1000)
```

```
INSERT INTO Student (studentId, residenceAdvisorId, strikes, rent) VALUES (12368295,
71849184, 0, 1100)
INSERT INTO Student (studentId, residenceAdvisorId, strikes, rent) VALUES (22385013,
81941048, 0, 1100)
INSERT INTO Student (studentId, residenceAdvisorId, strikes, rent) VALUES (44185018,
48105829, 0, 1000)
INSERT INTO Student (studentId, residenceAdvisorId, strikes, rent) VALUES (28492572,
71193810, 1, 750)

INSERT INTO MaintenanceRequest (requestId, studentId, staffId, description, status) VALUES
(74272757, 82957194, null, “broken dishwasher”, “submitted”)
INSERT INTO MaintenanceRequest (requestId, studentId, staffId, description, status) VALUES
(47827528, 71849184, 95282952, “air conditioning not working”, “assigned”)
INSERT INTO MaintenanceRequest (requestId, studentId, staffId, description, status) VALUES
(17492753, 44185018, null, “magnets on fridge fell off”, “complete”)
INSERT INTO MaintenanceRequest (requestId, studentId, staffId, description, status) VALUES
(15859324, 28492572, 18592520, “our light bulb broke”, “submitted”)
INSERT INTO MaintenanceRequest (requestId, studentId, staffId, description, status) VALUES
(82957295, 28492572, 18592520, “need a new chair for my room”, “submitted”)

INSERT INTO MaintenanceStaff (staffId, name, biweeklyWage)
VALUES (435214, “Jan Smailbegovic”, 2105)
INSERT INTO MaintenanceStaff (staffId, name, biweeklyWage)
VALUES (913037, “Aayush Behl”, 2067)
INSERT INTO MaintenanceStaff (staffId, name, biweeklyWage)
VALUES (323271, “Alex Briauzov”, 1999)
INSERT INTO MaintenanceStaff (staffId, name, biweeklyWage)
VALUES (837249, “Penny West”, 2450)
INSERT INTO MaintenanceStaff (staffId, name, biweeklyWage)
VALUES (174528, “Ben Stiller”, 2750)

INSERT INTO Sublet (studentId, residentId, name, startDate, endDate) VALUES (12345678,
28492572, “Jeffrey Lew”, “20 September 2024”, “1 December 2024”)
INSERT INTO Sublet (studentId, residentId, name, startDate, endDate) VALUES (82957194,
12368295, “Jack Reacher”, “1 September 2023”, “1 January 2024”)
INSERT INTO Sublet (studentId, residentId, name, startDate, endDate) VALUES (34568213,
13849148, “Matthew Scott”, “1 January 2024”, “1 December 2024”)
INSERT INTO Sublet (studentId, residentId, name, startDate, endDate) VALUES (15154466,
12368295, “Mathias Der”, “18 September 2022”, “31 December 2022”)
```

```
INSERT INTO Sublet (studentId, residentId, name, startDate, endDate) VALUES (78755003,
44185018, “Will Bjorn”, “20 September 2024”, “1 December 2024”)

INSERT INTO Package (id, studentId, mailbox, deliveryDate) VALUES (12842953, 82957194, 22,
“27 November 2024”)
INSERT INTO Package (id, studentId, mailbox, deliveryDate) VALUES (72957295, 12368295, 4,
“12 November 2024”)
INSERT INTO Package (id, studentId, mailbox, deliveryDate) VALUES (29492482, 44185018, 11,
“2 November 2024”)
INSERT INTO Package (id, studentId, mailbox, deliveryDate) VALUES (56275925, 13849148, 7,
“30 October 2024”)
INSERT INTO Package (id, studentId, mailbox, deliveryDate) VALUES (48592853, 44185018, 5, “2
November 2024”)
```