University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: __2_____

Date: _Mar 02, 2025_____

Group Number: ____110_____

Name	**Student Number**	**CS Alias (Userid)**	**Preferred E-mail Address**
Frank Yang	11753753	j6l4t	ffyang@student.ubc.ca
Xingyang Zheng	57446361	c5i2r	xingyang2027@gmail.com
Shiyu Zhou	27214782	z9y5k	szhou49@outlook.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

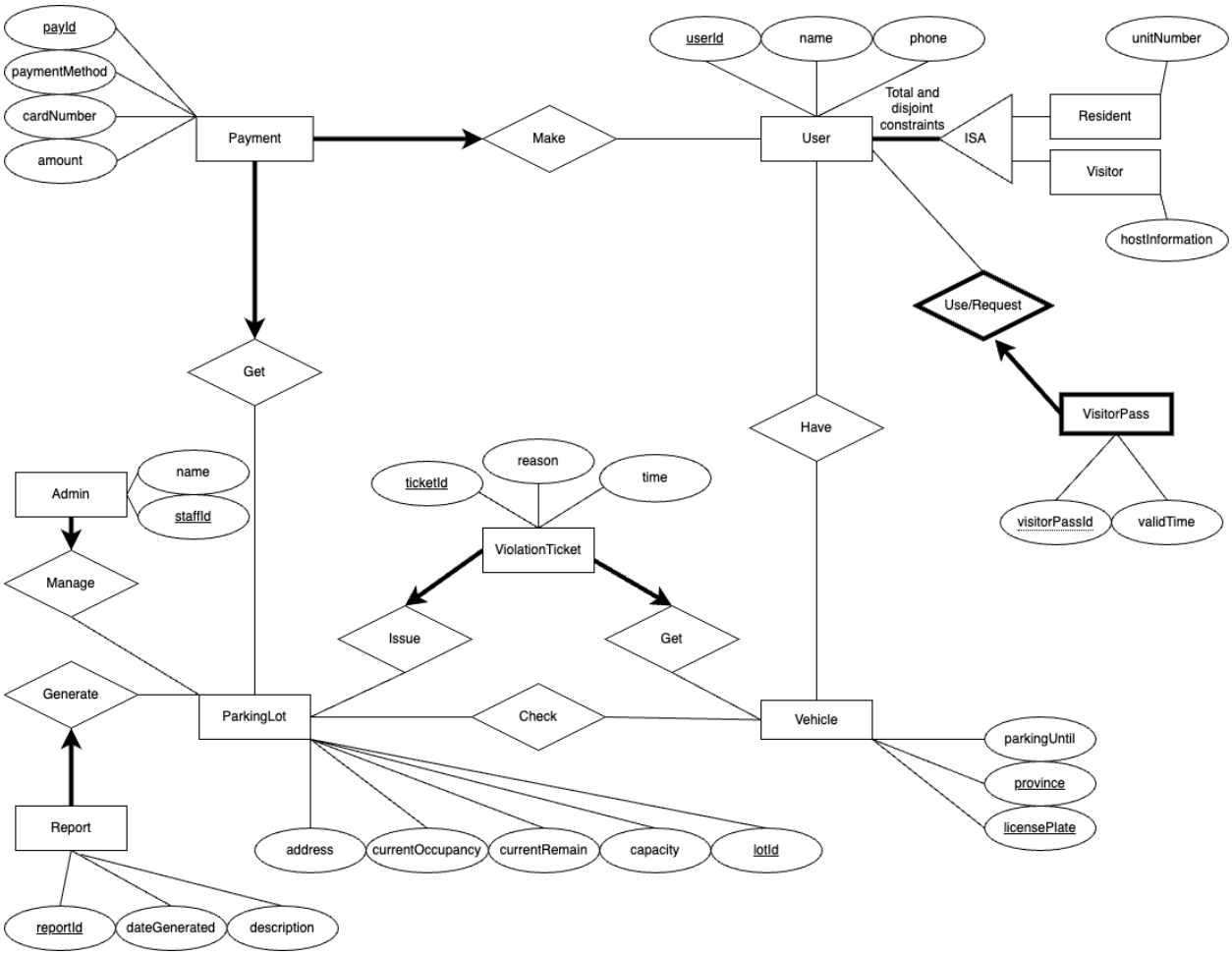
In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

University of British Columbia, Vancouver
Department of Computer Science

Project Summary:

This project focuses on **residential parking management**, addressing inefficiencies in traditional visitor pass systems. It ensures fair usage of visitor parking spaces by tracking parking activities, enforcing regulations, and preventing misuse. The system also provides administrators with tools to monitor violations, manage payments, and take enforcement actions when necessary.

ER diagram:



The ER diagram we are using is an updated version of our Milestone 1 submission, incorporating the following improvements:

1. Fixed ISA D&T convention issues.

2. Clarified relationships by explicitly defining that users make payments and admins manage parking lots, ensuring proper representation of system functionality.
3. Enhanced readability by improving diagram presentation.
4. Corrected a typo in visitor pass ID labeling.
5. Added paymentMethod and cardNumber attributes to the Payment entity to store transaction details.
6. Renamed State/Prov. to province for consistency in naming.
7. Added capacity, currentOccupancy, and currentRemain attributes to the ParkingLot entity to track space availability dynamically.

University of British Columbia, Vancouver
Department of Computer Science

Schema:

ParkingLot(lotId: INTEGER, address: VARCHAR[100], capacity: INTEGER, currentOccupancy: INTEGER, currentRemain: INTEGER)
Constraints
PK: lotId
CK: address
FK: NULL
must maintain:

1) from ERD:
   NOT NULL: NULL
   UNIQUE: NULL
2) from Business need:
   NOT NULL: address, capacity, currentOccupancy, currentRemain
   UNIQUE: address

User(userId: INTEGER, name: VARCHAR[20], phone: INTEGER）
PK: userId
CK: NULL
FK: NULL
must maintain:

1) from ERD:
   NOT NULL: NULL
   UNIQUE: NULL
2) from Business need:
   NOT NULL: name, phone
   UNIQUE: phone

Resident(**userId**: INTEGER, unitNumber: INTEGER)
PK: userId
CK: NULL
FK: userId
must maintain:

3) from ERD:
   NOT NULL: NULL
   UNIQUE: NULL
4) from Business need:
   NOT NULL: NULL
   UNIQUE: NULL

Visitor(**userId**: INTEGER, hostInformation: VARCHAR[100])

PK: userId
CK: NULL
FK: userId
must maintain:

5) from ERD:
   NOT NULL: NULL
   UNIQUE: NULL
6) from Business need:
   NOT NULL: NULL
   UNIQUE: NULL

Vehicle(province: CHAR[20], licensePlate: VARCHAR[20], parkingUntil: DATETIME)
PK:province, licensePlate
CK: NULL
FK: NULL
must maintain:

7) from ERD:
   NOT NULL: NULL
   UNIQUE: NULL
8) from Business need:
   NOT NULL: NULL
   UNIQUE: NULL

Admin manage ParkingLot:
Admin(staffId: INTEGER, name: VARCHAR[20], **lotId**: INTEGER)
PK: staffId
CK: NULL
FK: lotId
must maintain:

9) from ERD:
   NOT NULL: lotId
   UNIQUE: NULL
10) from Business need:
   NOT NULL: name
   UNIQUE: NULL

ParkingLot Generate Report:
Report(reportId: INTEGER, dateGenerated: DATETIME, description: VARCHAR[255], **lotId**: INTEGER)
PK: reportId
CK: NULL
FK: lodId
must maintain:

11) from ERD:
    NOT NULL: lotId
    UNIQUE: NULL
12) from Business need:
    NOT NULL: dateGenerated
    UNIQUE: NULL

ParkingLot issue ViolationTicket and Vehicle Get it:
ViolationTicket(ticketId: INTEGER, reason: VARCHAR[100], time: DATETIME, **lotId**: INTEGER, **province**: VARCHAR[20], **licensePlate**: VARCHAR[20])
PK: ticketId
CK: NULL
FK: lotId,province, licensePlate
must maintain:

13) from ERD:
    NOT NULL: licensePlate, province, lodId
    UNIQUE: NULL
14) from Business need:
    NOT NULL: reason, time
    UNIQUE: NULL

User make Payment:
Payment(payId: INTEGER, amount: INTEGER, paymentMethod: VARCHAR[20], cardNumber: INTEGER, **userId**: INTEGER, **lotId**: INTEGER)
PK: payId
CK: NULL
FK: userId, lotId
must maintain:

15) from ERD:
    NOT NULL: userId, loadId
    UNIQUE: NULL
16) from Business need:
    NOT NULL: paymentMethod, cardNumber, amount
    UNIQUE: NULL

User Use or Request VisitorPass:
VisitorPass(visitorPassId: INTEGER, validTime: DATETIME, **userId**: INTEGER)
PK: visitorPassId, userId
CK: NULL
FK: userId
must maintain:

17) from ERD:
    NOT NULL: NULL

UNIQUE: NULL

18) from Business need:
NOT NULL: validTime
UNIQUE: NULL

ParkingLot Check Vehicle:
Check(**lotId**: INTEGER, **province**: VARCHAR[20], **licensePlate**: VARCHAR[20])
PK: lotId, province, licensePlate
CK: NULL
FK: userId,province., licensePlate
All attributes are PRIMARY KEY, so they are all NOT NULL and UNIQUE

User Have Vehicle:
Have(**userId**: INTEGER, **province**: VARCHAR[20], **licensePlate**: VARCHAR[20])
PK: userId, province, licensePlate
CK: NULL
FK: userId, province, licensePlate
All attributes are PRIMARY KEY, so they are all NOT NULL and UNIQUE

University of British Columbia, Vancouver
Department of Computer Science

Functional Dependencies (FDs):

User:
userId -> name, phone
name, phone -> userId
Resident:
unitNumber -> userId
userId -> unitNumber
Visitor:
hostInformation -> userId
userId -> hostInformation
Resident:
userId -> name, phone, unitNumber
name, phone -> userId, unitNumber
Visitor:
userId -> name, phone, hostInformation
name, phone -> userId, hostInformation
ViolationTicket:
ticketId -> reason, time, lotId, province, licensePlate
lotId, province, licensePlate -> reason, time
time, province, licensePlate -> lotId
ParkingLot:
lotId -> address, capacity, currentOccupancy, currentRemain
address -> lotId
capacity, currentOccupancy -> currentRemain
currentRemain, currentOccupancy-> capacity
capacity, currentRemain -> currentOccupancy

Vehicle:
province, vehiclePlate -> parkingUntil
Admin:
staffId -> name, lotId
Report:
reportId -> dateGenerated, description, lotId
dateGenerated, lotId -> reportId, description
Payment:
payId -> amount, paymentMethod, cardNumber, userId, lotId
cardNumber-> paymentMethod
VisitorPass:
visitorPassId -> validTime, userId

University of British Columbia, Vancouver
Department of Computer Science

Normalization:

BCNF Decomposition of the ViolationTicket Relation

ViolationTicket(ticketId: INTEGER, reason: VARCHAR, time: DATETIME, **lotId**: INTEGER**, province**: VARCHAR, **licensePlate**: VARCHAR)

- FD1: ticketId -> reason, time, lotId, province, licensePlate
- FD2: lotId, province, licensePlate -> reason, time
- FD3: time, province, licensePlate -> lotId

Since in FD2,
(lotId,province,licensePlate)+={lotId,province,licensePlate,reason,time}, it follows that (lotId,province, licensePlate) is not a superkey of the ViolationTicket relation. Thus, the ViolationTicket relation is not in BCNF, and we need to decompose it.
Using FD2, we decompose the relation into:

ViolationTicket1(lotId, province, licensePlate, reason, time)
ViolationTicket2(ticketId, lotId, province, licensePlate)
Since both ViolationTicket1 and ViolationTicket2 are in BCNF, no further decomposition is required.

BCNF Decomposition of the Payment Relation

Given the Payment relation:

Payment(payId: INTEGER, amount: INTEGER, paymentMethod: VARCHAR, cardNumber: VARCHAR, userId: INTEGER, lotId: INTEGER)

- FD1: payId -> amount, paymentMethod, cardNumber, userId, lotId
- FD2: cardNumber-> paymentMethod

Since in FD2,

(cardNumber)+ = {cardNumber, paymentMethod},

it follows that cardNumber is not a superkey of the Payment relation. Thus, the Payment relation is not in BCNF, and we need to decompose it.

Using FD2, we decompose the relation into:

1. Payment1(payId, amount, cardNumber, userId, lotId)

2. Payment2(cardNumber, paymentMethod)

Since both Payment1 and Payment2 are in BCNF, no further decomposition is required.

BCNF Decomposition of the ParkingLot Relation

Given the ParkingLot relation:

ParkingLot(lotId: INTEGER, address: VARCHAR(100), capacity: INTEGER, currentOccupancy: INTEGER, currentRemain: INTEGER)

- **FD1:** lotId -> address, capacity, currentOccupancy, currentRemain
- **FD2:** address -> lotId
- **FD3:** capacity, currentOccupancy -> currentRemain
- **FD4:** currentRemain, currentOccupancy -> capacity
- **FD5:** capacity, currentRemain -> currentOccupancy

Since in **FD3**,

(capacity, currentOccupancy)+ = {capacity, currentOccupancy, currentRemain},

it follows that (capacity, currentOccupancy) is not a superkey of the ParkingLot relation. Thus, the ParkingLot relation is not in BCNF, and we need to decompose it.

Using FD3, we decompose the relation into:

1. ParkingLot1(<u>capacity</u>, <u>currentOccupancy</u>, currentRemain)
2. ParkingLot2(<u>lotId</u>, address, capacity, currentOccupancy)

Since both ParkingLot1 and ParkingLot2 are in BCNF, no further decomposition is required.

University of British Columbia, Vancouver
Department of Computer Science

SQL statements:

Explanation for ON DELETE / ON UPDATE Choices

We selectively use ON DELETE CASCADE in relationships where removing the parent row makes the child records obsolete. For example, when deleting a parking lot (ParkingLot2) automatically removes associated reports (Report) and admin assignments (Admin). In other cases, such as audit or historical tracking, we leave the default RESTRICT/NO ACTION so the child data remains intact unless explicitly deleted.

Regarding ON UPDATE, since Oracle does not support ON UPDATE CASCADE, so we decided to handle any necessary key updates at the application layer.

```
CREATE TABLE ParkingLot1(
        capacity INTEGER NOT NULL,
        currentOccupancy INTEGER NOT NULL,
        currentRemain INTEGER NOT NULL,
        PRIMARY KEY(capacity, currentOccupancy)
 )


CREATE TABLE ParkingLot2(
        lotId INTEGER PRIMARY KEY,
        address VARCHAR(100) NOT NULL UNIQUE,
        capacity INTEGER NOT NULL,
        currentOccupancy INTEGER NOT NULL,
        FOREIGN KEY (capacity, currentOccupancy)
        REFERENCES ParkingLot1(capacity, currentOccupancy)
 )

CREATE TABLE User(
        userId INTEGER PRIMARY KEY,
        name VARCHAR(20) NOT NULL,
        phone INTEGER UNIQUE NOT NULL
)

CREATE TABLE Resident(
        userId INTEGER PRIMARY KEY,
        unitNumber INTEGER,
        FOREIGN KEY(userId) references User(userId)
```

```
                ON DELETE CASCADE
)

CREATE TABLE Visitor(
        userId INTEGER PRIMARY KEY,
        hostInformation VARCHAR(100),
        FOREIGN KEY(userId) references User(userId)
                ON DELETE CASCADE
)


CREATE TABLE Vehicle(
        province VARCHAR(20),
        licensePlate VARCHAR(20),
        parkingUntil DATETIME,
        PRIMARY KEY(province, licensePlate)
)

CREATE TABLE Admin(
        staffId INTEGER PRIMARY KEY,
        name VARCHAR(20) NOT NULL,
        lotId INTEGER NOT NULL,
        FOREIGN KEY (lotId) references ParkingLot2(lotId)
                ON DELETE CASCADE
)

CREATE TABLE Report(
        reportId INTEGER PRIMARY KEY,
        dataGenerated DATETIME NOT NULL,
        description VARCHAR(255),
        lotId INTEGER NOT NULL,
        FOREIGN KEY (lotId) references ParkingLot2(lotId)
                ON DELETE CASCADE
)

CREATE TABLE ViolationTicket1(
        lotId INTEGER NOT NULL,
        province VARCHAR(20) NOT NULL,
        licensePlate VARCHAR(20) NOT NULL,
        reason VARCHAR(100) NOT NULL,
        time DATETIME NOT NULL,
        PRIMARY KEY(lotId, province, licensePlate, time),
        FOREIGN KEY (lotId)
```

```
                REFERENCES ParkingLot2(lotId)
                ON DELETE CASCADE,
        FOREIGN KEY (province, licensePlate)
                REFERENCES Vehicle(province, licensePlate)
                ON DELETE CASCADE
)
CREATE TABLE ViolationTicket2(
        ticketId INTEGER PRIMARY KEY,
        lotId INTEGER NOT NULL,
        province VARCHAR(20) NOT NULL,
        licensePlate VARCHAR(20) NOT NULL,
        FOREIGN KEY (lotId)
                REFERENCES ParkingLot2(lotId)
                ON DELETE CASCADE,
        FOREIGN KEY (province, licensePlate)
                REFERENCES Vehicle(province, licensePlate)
                ON DELETE CASCADE
)


CREATE TABLE Payment1(
        payId INTEGER PRIMARY KEY,
        amount INTEGER NOT NULL,
        cardNumber VARCHAR(20) NOT NULL,
        userId INTEGER NOT NULL,
        lotId INTEGER NOT NULL,
        FOREIGN KEY (cardNumber)
                REFERENCES Payment2(cardNumber)
                ON DELETE CASCADE,
        FOREIGN KEY (userId)
                REFERENCES User(userId)
                ON DELETE CASCADE,
        FOREIGN KEY (lotId)
                REFERENCES ParkingLot2(lotId)
                ON DELETE CASCADE
)

CREATE TABLE Payment2(
        cardNumber VARCHAR(20) PRIMARY KEY,
        paymentMethod VARCHAR(20) NOT NULL
)
```

```
CREATE TABLE VisitorPass(
        visitorPassId INTEGER,
        validTime DATETIME NOT NULL,
        userId INTEGER NOT NULL,
        PRIMARY KEY(visitorPassId, userId),
        FOREIGN KEY (userId)
                REFERENCES User(userId)
                ON DELETE CASCADE
)


-- The table name "Check" is a reserved SQL keyword.
-- To avoid conflicts, we have renamed it to "VehicleCheck" while maintaining the same functionality.

CREATE TABLE VehicleCheck(
        lotId INTEGER,
        province VARCHAR(20),
        licensePlate VARCHAR(20),
        PRIMARY KEY(lotId,province, licensePlate),
        FOREIGN KEY (lotId)
                REFERENCES ParkingLot2(lotId)
                ON DELETE CASCADE,
        FOREIGN KEY (province, licensePlate)
                REFERENCES Vehicle(province, licensePlate)
                ON DELETE CASCADE
)


CREATE TABLE Have(
        userId INTEGER,
        province VARCHAR(20),
        licensePlate VARCHAR(20),
        PRIMARY KEY(userId,province, licensePlate),
        FOREIGN KEY (userId)
                REFERENCES User(userId)
                ON DELETE CASCADE,
        FOREIGN KEY (province, licensePlate)
                REFERENCES Vehicle(province, licensePlate)
                ON DELETE CASCADE
)
```

```
INSERT
INTO ParkingLot1(capacity, currentOccupancy, currentRemain)
VALUES (300, 150, 150),
        (100, 50, 50),
        (200, 110, 90),
        (700, 200, 500),
        (600, 150, 450);

INSERT
INTO
ParkingLot2(lotId, address, capacity, currentOccupancy)
VALUES (1, 'B1T1Z1, vancouver', 300, 150),
        (2, 'B3T4Z1, vancouver', 100, 50),
        (3, 'B2T9Z3, vancouver', 200, 110),
        (4, 'B9T2Z1, vancouver', 700, 200),
        (5, 'B8T3Z1, vancouver', 600, 150);

INSERT
INTO Payment1(payId, amount, cardNumber, userId, lotId)
VALUES (1, 100, '1111111111', 1, 1),
        (2, 50, '2222222222', 2, 2),
        (3, 120, '3333333333', 3, 1),
        (4, 80, '4444444444', 4, 3),
        (5, 200, '5555555555', 5, 1);

INSERT
INTO Payment2(cardNumber, paymentMethod)
VALUES ('1111111111', 'CreditCard'),
        ('2222222222', 'Debit'),
        ('3333333333', 'PayPal'),
        ('4444444444', 'MasterCard'),
        ('5555555555', 'Visa');


INSERT
INTO User(userId, name, phone)
VALUES (1, 'Alice', 1234567890),
        (2, 'Bob', 2345678901),
        (3, 'Charlie', 3456789012),
        (4, 'Amy', 22222222222),
        (5, 'Mary',1122334455),
        (6, 'Tom', 2748261538),
        (7, 'Ace', 8426387462),
```

```
        (8, 'Michael', 3527427384),
        (9, 'Oliver', 6452748262),
        (10, 'Ada', 7638472637);

INSERT
INTO Resident(userId, unitNumber)
VALUES (1, 101),
        (3, 105),
        (4, 208),
        (7, 709),
        (9,503);

INSERT
INTO Visitor(userId, hostInformation)
VALUES (2, 'visit 101'),
        (5, 'visit 208'),
        (6, 'visit 105'),
        (8, 'visit 709'),
        (10, 'visit 503');


INSERT
INTO Vehicle(province, licensePlate, parkingUntil)
VALUES ('BC', 'ABC123', '2025-06-01 12:00:00'),
        ('ON', 'XYZ789', '2025-07-15 18:30:00'),
        ('QC', 'LMN456', '2025-08-10 09:45:00'),
        ('AB', 'DEF234', '2025-09-20 14:00:00'),
        ('MB', 'GHI567', '2025-10-05 08:15:00'),
        ('SK', 'JKL890', '2025-11-30 23:59:59'),
        ('NS', 'MNO123', '2025-12-25 07:30:00'),
        ('NB', 'PQR456', '2026-01-01 12:00:00'),
        ('NL', 'STU789', '2026-02-14 15:45:00'),
        ('BC', 'VWX234', '2026-03-17 10:10:00'),
        ('ON', 'YZA567', '2026-04-22 11:30:00'),
        ('ON', 'BCD890', '2026-05-05 17:00:00'),
        ('SK', 'EFG123', '2026-06-18 20:20:00');

INSERT
INTO Admin(staffId, name, lotId)
VALUES (1, 'Ada', 1),
        (2, 'Oliver', 1),
        (3, 'Michael', 2),
        (4, 'Mary', 3),
```

```
        (5, 'Hason', 4);

INSERT
INTO Report(reportId, dataGenerated, description, lotId)
VALUES (1, '2025-02-28 10:30:00', 'Monthly parking lot usage report', 1),
        (2, '2025-03-01 08:45:00', 'Quarterly maintenance report', 2),
        (3, '2025-03-02 12:15:00', 'Weekly traffic analysis', 1),
        (4, '2025-03-03 09:00:00', 'Daily revenue report', 3),
        (5, '2025-03-03 11:20:00', 'Security incident report', 4);



INSERT
INTO ViolationTicket1(lotId, province, licensePlate, reason, time)
 VALUES (1, 'BC', 'ABC123', 'Parking in a restricted area', '2025-03-01 08:30:00'),
        (2, 'BC', 'XYZ789', 'Expired parking meter', '2025-03-02 09:45:00'),
        (1, 'ON', 'DEF456', 'Blocking emergency lane', '2025-03-02 11:15:00'),
        (3, 'AB', 'GHI321', 'No valid parking permit', '2025-03-03 14:00:00'),
        (4, 'BC', 'JKL987', 'Parking in handicapped space', '2025-03-03 16:20:00');

INSERT
INTO ViolationTicket2(ticketId, lotId, province, licensePlate)
VALUES (1, 1, 'BC', 'ABC123'),
        (2, 2, 'BC', 'XYZ789'),
         (3, 1, 'ON', 'DEF456'),
        (4, 3, 'AB', 'GHI321'),
        (5, 4, 'BC', 'JKL987');



INSERT
INTO VisitorPass (visitorPassId, validTime, userId)
VALUES (1, '2025-06-03 14:45:00', 3),
        (2, '2025-06-01 10:00:00', 1),
        (3, '2025-06-07 22:45:00', 7),
        (4, '2025-06-04 16:00:00', 4),
        (5, '2025-06-09 09:15:00', 9),
        (6, '2025-06-05 18:15:00', 5),
        (7, '2025-06-08 08:00:00', 8),
        (8, '2025-06-06 20:30:00', 6),
        (9, '2025-06-10 11:45:00', 10),
        (10, '2025-06-02 12:30:00', 2);
```

-- The table name "Check" is a reserved SQL keyword.
-- To avoid conflicts, we have renamed it to "VehicleCheck" while maintaining the same functionality.

```
INSERT
INTO VehicleCheck (lotId, province, licensePlate)
VALUES (1, 'BC', 'ABC123'),
        (2, 'BC', 'XYZ789'),
        (1, 'ON', 'DEF456'),
        (3, 'AB', 'GHI321'),
        (4, 'BC', 'JKL987');



INSERT INTO Have (userId, province, licensePlate)
VALUES  (1, 'BC', 'ABC123'),
        (2, 'ON', 'XYZ789'),
        (3, 'QC', 'LMN456'),
        (4, 'AB', 'DEF234'),
        (5, 'MB', 'GHI567'),
        (6, 'SK', 'JKL890'),
        (7, 'NS', 'MNO123'),
        (8, 'NB', 'PQR456'),
        (9, 'NL', 'STU789'),
        (10, 'BC', 'VWX234'),
        (3, 'ON', 'YZA567'),
        (5, 'ON', 'BCD890'),
        (7, 'SK', 'EFG123');
```

University of British Columbia, Vancouver
Department of Computer Science

Acknowledgment of use of AI tools:

Yes, we used ChatGPT to generate sample data for the INSERT statements in our SQL scripts. All generated data was reviewed and validated before inclusion.