University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: 2025-03-03

Group Number: 36

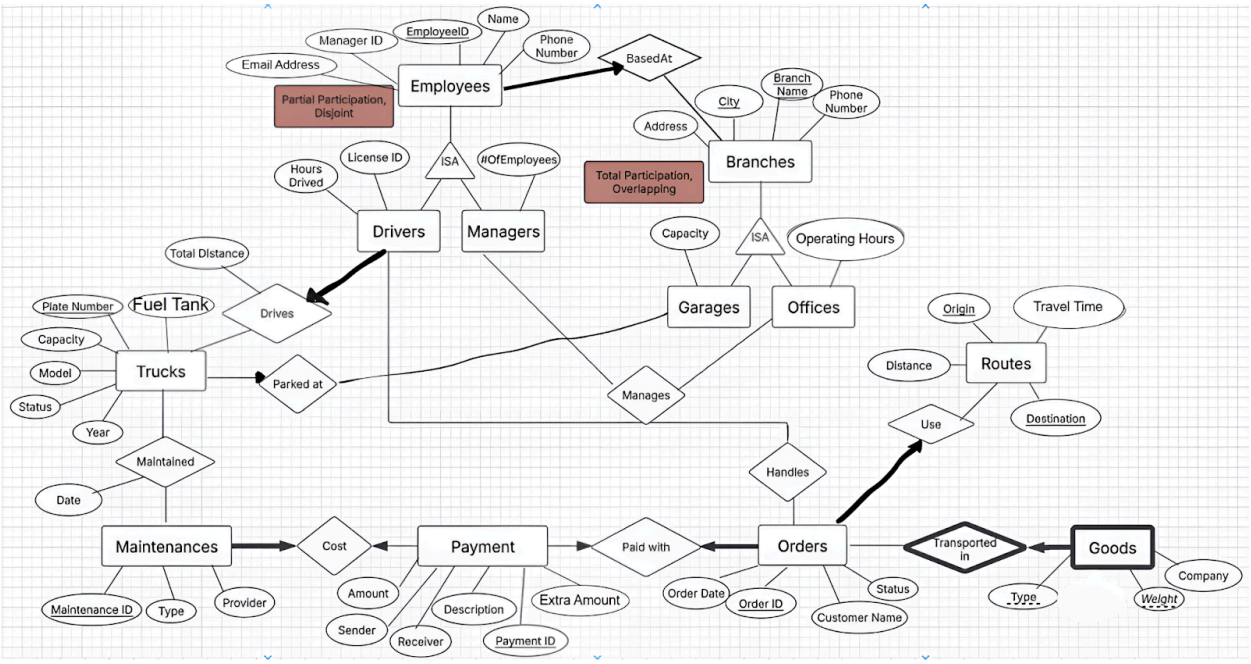
Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Randeep Sidhu	40853848	w9y9b@ugrad.cs.ubc.a	rsdihu33@student.ubc.ca
Richard Adhika	28751741	u8b2e@ugrad.cs.ubc.ca	radhik11@student.ubc.ca
Tegvir Multani	49064660	k4j6b@ugrad.cs.ubc.ca	tmultani@student.ubc.ca

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

2. Our project is about a trucking company, where we manage drivers, trucks and their maintenance, as well as the goods to be transported. The main action revolves around the Orders entity, where orders are managed with their goods, a specific route and a designated driver along with its according payment. Furthermore, we also have managers supervising the employees and offices, in which the company is run.

3. ER Diagram



Changes made:

- Added ManagerID attribute to Employees entity and BasedAt relation to add a non-trivial FD.
- Added EmailAddress attribute to Employees entity to have more candidate keys.
- Added FuelTank attribute to Trucks entity to add a non-trivial FD.
- Added Manages relation from Managers to Offices.
- Fixed the dashed line of the attributes in Goods entity.
- Fixed the arrow line issue for the Drives and Use relations.
- Renamed AverageTravelTime attribute in Routes entity to TravelTime for simplicity.
- Renamed Tips/Bonus/Penalties attribute in Payment entity to ExtraAmount for simplicity.
- Renamed CustomerServiceTime attribute in OperatingHours entity.
- Moved OrderDate attribute from the Handles relation to Orders Entity to accommodate for the change in the Handles relation.
- Removed Trucks entity from Handles relation since by knowing the driver, we also know the truck used.
- Added participation constraint and many-to-one relationship to Drives relation.

4. Schema

EmployeesBasedAt(EmployeeID:char(8), Name:varchar(30), PhoneNumber:char(10), EmailAddress:varchar(30), ManagerID:char(8), BranchName:varchar(30), City:varchar(20))
PK:(EmployeeID)
CK:(PhoneNumber), ({Name, EmailAddress})
FK:({BranchName, City}) References Branches
Not Null: (Name, PhoneNumber)
Unique:(PhoneNumber)

DriversDrive(EmployeeID:char(8), HoursDrived:int, LicenseID:char(10), TotalDistance:int, PlateNumber:char(6))
PK:(EmployeeID)
CK:(LicenceID)
FK:(EmployeeID) References Employees
FK:(PlateNumber) References TrucksParkedAt
Not Null:(LicenseID)
Unique:(LicenseID)

Managers(EmployeeID:char(8), #OfEmployees:int)
PK:(EmployeeID)
FK:(EmployeeID) References Employees

Manages(EmployeeID:char(8), BranchName:varchar(30), City:varchar(20))
PK:({BranchName, City, EmployeeID})
FK:({BranchName, City}) References Offices
FK:(EmployeeID) References Managers

Branches(BranchName:varchar(30), City:varchar(20), Address:varchar(30), PhoneNumber:char(10))
PK:({BranchName, City})
CK:(PhoneNumber, {BranchName, Address})
Not Null: PhoneNumber
Unique: PhoneNumber

Garages(BranchName:varchar(30), City:varchar(20), Capacity:int)
PK:({BranchName, City})
FK:({BranchName, City}) References Branches
Not Null: Capacity

Offices(BranchName:varchar(30), City:varchar(20), OperatingHours: char(12))
PK:({BranchName, City})
FK:({BranchName, City}) References Branches
Not Null: OperatingHours

TrucksParkedAt(PlateNumber:char(6), Capacity:int, Model:char(5), Status:varchar(10), Year:char(4), FuelTank:int, BranchName:varchar(30), City:varchar(20))
PK:(PlateNumber)
FK:({BranchName, City}) References Garages
Not Null: Capacity, Status

Routes(Origin:varchar(40), Destination:varchar(40), TravelTime:int, Distance:int)
PK:({Origin,Destination})

OrdersUsedPaid(OrderID:char(8), OrderDate:date, CustomerName:varchar(30), Status:varchar(10), Origin:varchar(30), Destination:varchar(30), PaymentID:char(8))
PK:(OrderID)
FK:({Origin,Destination}) References Routes
FK: (PaymentID) References PaymentsPaid
Unique: (PaymentID)
Not Null: (PaymentID, Origin, Destination)

Handles(OrderID:char(8), EmployeeID:char(8))
PK({OrderID, EmployeeID})
FK:(OrderID) References OrdersUsedPaid
FK:(EmployeeID) References DriversDrive

TransportedGoods(Weight:int, Company:varchar(30), Type:varchar(15), OrderID:char(8))
PK:({OrderID, Type, Weight})
FK:(OrderID) References OrdersUsed

Payments(PaymentID:char(8), Description:char(20), Receiver:char(8), Sender:char(8), Amount:int, ExtraAmount:int OrderID:char(8), MaintenanceID:char(8))
PK:(PaymentID)
FK:(OrderID) References OrdersUsedPaid
FK:(MaintenanceID) References MaintenancesCost
Unique(OrderID)
Unique(Maintenance)

MaintenancesCost(MaintenanceID:char(8), Type:varchar(15), Provider:varchar(30), PaymentID:char(8))
PK:(MaintenanceID)
FK:(PaymentID) References Payments
Unique(PaymentID)
Not Null(PaymentID)

Maintained(Date:date, PlateNumber:char(6), MaintenanceID:char(8))
PK:({MaintenanceID, PlateNumber})
FK:(MaintenanceID) References Maintenances
FK:(PlateNumber) References Trucks

5, 6) Functional Dependencies (FD) and BCNF Decomposition

EmployeesBasedAt
FD:
- EmployeeID -> EmailAddress, ManagerID, Name, PhoneNumber, BranchName, City
- PhoneNumber -> EmailAddress, ManagerID, Name, EmployeeID, BranchName, City
- ManagerID -> BranchName, City

BCNF Decomposition:
Not in BCNF due to the last FD (PhoneNumber is not a superkey).
Decomposed to EmployeesBasedAt1(ManagerID, **BranchName, City)** and
EmployeesBasedAt2(EmployeeID, EmailAddress, Name, PhoneNumber, **ManagerID**).

DriversDrive
FD:
- EmployeeID -> LicenseID, HoursDrived, PlateNumber
- LicenseID-> EmployeeID, HoursDrived, PlateNumber

BCNF Decomposition: Already in BCNF.

Managers
FD:
- EmployeeID -> #OfEmployees

BCNF Decomposition: Already in BCNF.

Manages
FD: no non-trivial FDs.
BCNF Decomposition: Already in BCNF.

Branches
FD:
- BranchName, City -> Address, PhoneNumber
- PhoneNumber -> BranchName, City, Address
- BranchName, Address-> City, PhoneNumber

BCNF Decomposition: Already in BCNF.

Garages
FD:
- BranchName, City -> Capacity

BCNF Decomposition: Already in BCNF

Offices
FD:

- BranchName, City -> OperatingHours

BCNF Decomposition: Already in BCNF.

TrucksParkedAt
FD:

- PlateNumber -> Capacity, Model, Status, Year, FuelTank, BranchName, City
- Model -> Capacity, FuelTank

BCNF Decomposition:
Not in BCNF due to the last FD (Model is not a superkey).
Decomposed to TrucksParkedAt1(Model, Capacity, FuelTank) and
TrucksParkedAt2(PlateNumber,Status, Year, **BranchName, City, Model**).

Routes
FD:

- Origin, Destination -> Distance, TravelTime
- Distance -> TravelTime

BCNF Decomposition:
Not in BCNF due to the last FD (Distance is not a superkey).
Decomposed to Routes1(Distance, TravelTime) and Routes2(**Distance**, Origin, Destination).

OrdersUsedPaid
FD:

- OrderID -> CustomerName, Status, Origin, Destination, OrderDate, PaymentID

BCNF Decomposition: Already in BCNF.

Handles
FD: no non-trivial FD
BCNF Decomposition: Already in BCNF.

TransportedGoods
FD:

- Type, Weight, OrderID -> Company

BCNF Decomposition: Already in BCNF.

Payments
FD:

- PaymentID-> Description, Receiver, Sender, Amount, ExtraAmount, OrderID, MaintenanceID
- Description -> ExtraAmount

BCNF Decomposition:
Not in BCNF due to the last FD (Description is not a superkey).
Decomposed to Payments1(Description, ExtraAmount) and Payments2(PaymentID, Receiver, Sender, Amount, **Description**, OrderID, MaintenanceID).

MaintenancesCost
FD:

- MaintenanceID -> Type, Provider, PaymentID

BCNF Decomposition: Already in BCNF.

Maintained
FD:

- PlateNumber, MaintenanceID -> Date

BCNF Decomposition: Already in BCNF.

7) SQL/DDL Statements to Create the Table

```
CREATE TABLE EmployeesBasedAt1(
        ManagerID CHAR(8),
        BranchName VARCHAR(30),
        City VARCHAR(20),
        PRIMARY KEY (ManagerID),
        FOREIGN KEY (BranchName, City) REFERENCES Branches(BranchName, City)
                ON DELETE SET NULL
                ON UPDATE CASCADE
)
```

If a branch is removed, it does not make sense to remove all its corresponding employer, and thus while waiting for allocation to another branch, we can just set it to NULL. If a branch's name or city is updated, then it makes sense to also update it in the corresponding employee table.

```
CREATE TABLE EmployeesBasedAt2(
        EmployeeID CHAR(8),
        Name VARCHAR(30) NOT NULL,
        PhoneNumber CHAR(10) UNIQUE NOT NULL,
        EmailAddress VARCHAR(30),
        ManagerID CHAR(8),
        PRIMARY KEY (EmployeeID),
        FOREIGN KEY (ManagerID) REFERENCES EmployeesBasedAt1(ManagerID)
                ON DELETE SET NULL
                ON UPDATE CASCADE
)
```

If a manager is removed, it does not make sense to remove all his corresponding employees, and thus while waiting for allocation to another manager, we can just set it to NULL. If a manager's id is updated, then it makes sense to also update it in the corresponding employee table.

```
CREATE TABLE DriversDrive(
        EmployeeID CHAR(8),
        HoursDrived INT
        LicenseID CHAR(10) UNIQUE NOT NULL,
        TotalDistance INT,
        PlateNumber CHAR(6),
        PRIMARY KEY (EmployeeID),
        FOREIGN KEY (EmployeeID) REFERENCES EmployeesBasedAt2(EmployeeID)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
```

```
    FOREIGN KEY (PlateNumber) REFERENCES TrucksParkedAt2(PlateNumber)
        ON DELETE SET NULL
        ON UPDATE CASCADE
)
```

If a truck is removed, then we don't want to remove its corresponding driver, rather we set it to null to indicate that the driver is not associated to any truck at the moment. Since a driver is an employee, we have to remove the driver if we remove the employee. Furthermore, the driver's id will also be updated if the employee's id is updated since they are essentially the same thing.

```
CREATE TABLE Managers(
    EmployeeID CHAR(8),
    #OfEmployees INT,
    PRIMARY KEY (EmployeeID)
    FOREIGN KEY (EmployeeID) REFERENCES EmployeesBasedAt2(EmployeeID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
)
```

Since a manager is an employee, we have to remove the manager if we remove the employee. Furthermore, the manager's id will also be updated if the employee's id is updated since they are essentially the same thing.

```
CREATE TABLE Manages(
    EmployeeID CHAR(8),
    BranchName VARCHAR(30),
    City VARCHAR(20),
    PRIMARY KEY (EmployeeID, BranchName, City),
    FOREIGN KEY (EmployeeID) REFERENCES Managers(EmployeeID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
    FOREIGN KEY(BranchName, City) REFERENCES Offices(BranchName, City)
        ON DELETE CASCADE
        ON UPDATE CASCADE
)
```

If a manager is removed, we have to remove the relation between all managers managing that office with themselves. The same thing applies if an office is removed. If a manager's key or office's key is updated, then it makes sense to also update it in the corresponding manages table.

```
CREATE TABLE Branches(
        BranchName VARCHAR(30),
        City VARCHAR(20),
        Address VARCHAR(30),
        PhoneNumber CHAR(10) UNIQUE NOT NULL,
        PRIMARY KEY (BranchName, City)
)

CREATE TABLE Garages(
        BranchName VARCHAR(30),
        City VARCHAR(20),
        Capacity INT NOT NULL,
        PRIMARY KEY (BranchName, City),
        FOREIGN KEY(BranchName, City) REFERENCES Branches(BranchName, City)
                ON DELETE CASCADE
                ON UPDATE CASCADE
)
```

Since a garage is a branch, we have to remove the garage if we remove the branch. Furthermore, the garage's key will also be updated if the branch's key is updated since they are essentially the same thing.

```
CREATE TABLE Offices(
        BranchName VARCHAR(30),
        City VARCHAR(20),
        OperatingHours CHAR(12) NOT NULL,
        PRIMARY KEY (BranchName, City),
        FOREIGN KEY(BranchName, City) REFERENCES Branches(BranchName, City)
                ON DELETE CASCADE
                ON UPDATE CASCADE
)
```

Since an office is a branch, we have to remove the office if we remove the branch. Furthermore, the office's key will also be updated if the branch's key is updated since they are essentially the same thing.

```
CREATE TABLE TrucksParkedAt1(
        Model CHAR(5),
        FuelTank INT,
        Capacity INT NOT NULL,
        PRIMARY KEY (Model)
```

```
)


CREATE TABLE TrucksParkedAt2(
        PlateNumber CHAR(6),
        Model CHAR(5),
        Status VARCHAR(10) NOT NULL,
        Year CHAR(4),
        BranchName VARCHAR(30),
        City VARCHAR(20),
        PRIMARY KEY (PlateNumber),
        FOREIGN KEY(BranchName, City) REFERENCES Garage(BranchName, City)
                ON DELETE SET NULL
                ON UPDATE CASCADE
        FOREIGN KEY(Model) REFERENCES TrucksParkedAt1(Model)
                ON DELETE SET NULL
                ON UPDATE CASCADE
)
```

If a garage is removed, it does not make sense to remove all the trucks parked there, and thus while waiting for allocation to another garage, we can just set it to NULL. If a model of truck is deleted (though realistically almost never happen), we can also just set it to NULL since it does not make sense to remove the truck directly. As usual, if the garage's key or model's key is updated, then we can also update the corresponding value in this table.

```
CREATE TABLE Routes1(
        Distance INT,
        TravelTime INT,
        PRIMARY KEY (Distance)
)




CREATE TABLE Routes2(
        Origin VARCHAR(40),
        Destination VARCHAR(40),
        Distance INT,
        PRIMARY KEY (Origin, Destination)
        FOREIGN KEY (Distance) REFERENCES Routes1(Distance)
                ON DELETE SET NULL
                ON UPDATE SET NULL
)
```

If a distance is deleted, then we can still keep the routes since distance and time travel of a route is not that crucial. Furthermore, if the distance is updated, then we also want the distance to be NULL since it does not make sense for the distance of two locations to change.

```
CREATE TABLE OrdersUsedPaid(
        OrderID CHAR(8),
        OrderDate DATE,
        CustomerName VARCHAR(30),
        Status VARCHAR(10),
        Origin VARCHAR(30) NOT NULL,
        Destination VARCHAR(30) NOT NULL,
        PaymentID CHAR(8) UNIQUE NOT NULL,
        PRIMARY KEY (OrderID),
        FOREIGN KEY(Origin, Destination) REFERENCES Routes2(Origin, Destination)
                ON DELETE SET NULL
                ON UPDATE CASCADE,
        FOREIGN KEY(PaymentID) REFERENCES Payments2(PaymentID)
                ON DELETE CASCADE
                ON UPDATE CASCADE
)
```

If a route is deleted, then even if we can't proceed with the shipment, we can still wait until a new corresponding route is established (probably by confirming it with the customer) rather than just removing the order completely, so we should set to null here. However, if a payment is removed, we should remove the order because customers have to pay to use our service. If the Route's key or Payment's key is updated, we can just reflect the change here.

```
CREATE TABLE Handles(
        OrderID CHAR(8),
        EmployeeID CHAR(8),
        PRIMARY KEY (OrderID, EmployeeID),
        FOREIGN KEY(EmployeeID) REFERENCES DriversDrive(EmployeeID)
                ON DELETE SET NULL
                ON UPDATE CASCADE,
        FOREIGN KEY(OrderID) REFERENCES OrdersUsedPaid(OrderID)
                ON DELETE CASCADE
                ON UPDATE CASCADE
)
```

If a driver is deleted, then we can assign another driver to an order rather than just removing the order, so we will set it to null first. However, if the order is removed, then there is nothing to handle so we will also delete the handle relation. If the Employee's key or Order's key is updated, we can just reflect the change here.

```
CREATE TABLE TransportedGoods(
        Weight INT,
        Company VARCHAR(30),
        Type VARCHAR(15),
        OrderID CHAR(8),
        PRIMARY KEY (Weight, Type, OrderID),
        FOREIGN KEY(OrderID) REFERENCES OrdersUsedPaid(OrderID)
                ON DELETE CASCADE
                ON UPDATE CASCADE
)
```

Note that the good here is a weak entity of order. Hence if an order is removed then there is nothing to be shipped, so we would also remove the goods. If the Order's key is updated, we can just reflect the change here.

```
CREATE TABLE Payments1(
        Description VARCHAR(100),
        ExtraAmount INT,
        PRIMARY KEY (Description)
)

CREATE TABLE Payments2(
        PaymentID CHAR(8),
        Description VARCHAR(100),
        Receiver VARCHAR(30),
        Sender VARCHAR(30),
        Amount INT,
        OrderID CHAR(8)
        MaintenanceID CHAR(8)
        PRIMARY KEY (PaymentID)
        FOREIGN KEY (OrderID) REFERENCES OrderUsedPaid (OrderID)
                ON DELETE CASCADE
                ON UPDATE CASCADE
        FOREIGN KEY (MaintenanceID) REFERENCES MaintenancesCost (MaintenanceID)
                ON DELETE CASCADE
                ON UPDATE CASCADE
        FOREIGN KEY (Description) REFERENCES Payment1(Description)
                ON DELETE SET NULL
                ON UPDATE CASCADE
)
```

If a description is deleted, then we can just set it to null since the payment is not much very much affected by it (it is not an important attribute). However, if an order or a maintenance it is

referring to is deleted, then it doesn't make sense to have a payment related to them anymore. As usual, any change to the foreign key will also follow here.

```
CREATE TABLE MaintenancesCost (
        MaintenanceID CHAR(8),
        Type VARCHAR(15),
        Provider VARCHAR(30),
        PaymentID CHAR(8),
        PRIMARY KEY (MaintenanceID),
        FOREIGN KEY(PaymentID) REFERENCES Payments2(PaymentID)
                ON DELETE CASCADE
                ON UPDATE CASCADE
)
```

Similar case to before, a maintenance must have a payment associated to it, so if the payment is removed, we should also remove the maintenance as it is no longer relevant. As usual, any change to the foreign key will also follow here.

```
CREATE TABLE Maintained (
        Date DATE,
        PlateNumber CHAR(6),
        MaintenanceID CHAR(8),
        PRIMARY KEY (MaintenanceID, PlateNumber),
        FOREIGN KEY(MaintenanceID) REFERENCES MaintenancesCost(MaintenanceID)
                ON DELETE CASCADE
                ON UPDATE CASCADE
        FOREIGN KEY(PlateNumber) REFERENCES TrucksParkedAt2(PlateNumber)
                ON DELETE CASCADE
                ON UPDATE CASCADE
 )
```

If a truck or its corresponding maintenance is removed, then the fact that the truck is being maintained at that particular point is no longer relevant, so we have to remove it as well. As usual, any change to the foreign key will also follow here.

8) INSERT Statements to Populate the Table

EmployeesBasedAt1:

```
INSERT INTO EmployeesBasedAt1
 VALUES ('MNG10000', 'Fleetwood', 'Surrey')

INSERT INTO EmployeesBasedAt1
```

VALUES (‘MNG10001’, ‘Newton’, ‘Surrey’)

INSERT INTO *EmployeesBasedAt1*
VALUES (‘MNG10002’, ‘Dunbar’, ‘Vancouver’)

INSERT INTO *EmployeesBasedAt1*
VALUES (‘MNG10003’, ‘Ladner’, ‘Delta’)

INSERT INTO *EmployeesBasedAt1*
VALUES (‘MNG10004’, ‘Downtown’, ‘Vancouver’)

EmployeesBasedAt2:

INSERT INTO DriversBa*EmployeesBasedAt2*sedAt
VALUES (‘EMP50000’, ‘Kabir’, ‘6041033324’, ‘kabir@truckltd.com’, ‘MNG10000’)

INSERT INTO EmployeesBasedAt2
VALUES (‘EMP50001’, ‘Ryan’, ‘6044248234’, ‘ryan@truckltd.com’, ‘MNG10001’)

INSERT INTO EmployeesBasedAt2
VALUES (‘EMP50002’, ‘James’, ‘604531342’, ‘james@truckltd.com’, ‘MNG10002’)

INSERT INTO EmployeesBasedAt2
VALUES (‘EMP50003’, ‘Japneet’, ‘6049489991’, ‘japneet@truckltd.com’, ‘MNG10003’)

INSERT INTO EmployeesBasedAt2
VALUES (‘EMP50004’, ‘Alisha’, ‘6042343312’, ‘alisha@truckltd.com’, ‘MNG10003’)

INSERT INTO EmployeesBasedAt2
VALUES (‘MNG00000’, ‘Hans R’, ‘6042342324’, ‘hansruther@truckltd.com’, null)

INSERT INTO EmployeesBasedAt2
VALUES (‘MNG00001’, ‘Michael’, ‘6042314302’, ‘michael1990@truckltd.com’, null)
INSERT INTO EmployeesBasedAt2
VALUES (‘MNG00002’, ‘Jennifer’, ‘6042312391’, ‘jen11@truckltd.com’, ‘MNG00000’)

INSERT INTO EmployeesBasedAt2
VALUES (‘MNG00003’, ‘Leonard’, ‘6042123511’, ‘lenoard@truckltd.com’, null)

INSERT INTO EmployeesBasedAt2
VALUES (‘MNG00004’, ‘Paresh’, ‘6042314921’, ‘paresh71@truckltd.com’, ‘MNG00001’)

DriversDrive:

INSERT INTO DriversDrive
VALUES (‘EMP50000’, 80, ‘LICENSE100’, 100000, ‘TRU100’)

INSERT INTO DriversDrive
VALUES (‘EMP50001’, 90, ‘LICENSE101’, 85000, ‘TRU101’)

INSERT INTO DriversDrive
VALUES (‘EMP50002’, 70, ‘LICENSE102’, 940000, ‘TRU102’)

INSERT INTO DriversDrive
VALUES (‘EMP50003’, 110, ‘LICENSE103’, 80000, ‘TRU103’)

INSERT INTO DriversDrive
VALUES (‘EMP50004’, 140, ‘LICENSE104’, 122000, ‘TRU104’)

Managers:

INSERT INTO Managers
VALUES (‘MNG10000’, 1)

INSERT INTO Managers
VALUES (‘MNG10001’, 1)

INSERT INTO Managers
VALUES (‘MNG10002’, 1)

INSERT INTO Managers
VALUES (‘MNG10003’, 2)

INSERT INTO Managers
VALUES (‘MNG10004’, 0)

Manages:

INSERT INTO Manages
VALUES ('EMP12000', ‘Fleetwood’, ‘Surrey'');

INSERT INTO Manages
VALUES ('EMP12001', ‘Newton’, ‘Surrey’);

INSERT INTO Manages
VALUES ('EMP12002', ‘Dunbar’, ‘Vancouver’);

INSERT INTO Manages
VALUES ('EMP12003', ‘Ladner’, ‘Delta’);

INSERT INTO Manages
VALUES ('EMP12004', ‘Downtown’, ‘Vancouver’);

Branches:

INSERT INTO Branches
VALUES (‘Fleetwood’, ‘Surrey'', ‘123 Way’, ‘6042332123’);

INSERT INTO Branches
VALUES (‘Newton’, ‘Surrey’, ‘3232 Milk View’, ‘7783323321’);

INSERT INTO Branches
VALUES (‘Dunbar’, ‘Vancouver’, ‘421 Marine Drive’, ‘7781234291’);

INSERT INTO Branches
VALUES (‘Ladner’, ‘Delta’, ‘82331 Pine Street’, ‘4213413332’);

INSERT INTO Branches
VALUES (‘Downtown’, ‘Vancouver’, ‘233 Red Way’, ‘6472331239’);

INSERT INTO Branches
VALUES (‘Westwood’, ‘Surrey'', ‘123 Way’, ‘6042332123’);

INSERT INTO Branches
VALUES (‘East’, ‘Surrey’, ‘3232 Milk View’, ‘7783323321’);

INSERT INTO Branches
VALUES (‘Ridaeu’, ‘Vancouver’, ‘421 Marine Drive’, ‘7781234291’);

INSERT INTO Branches
VALUES (‘Cader’, ‘Delta’, ‘82331 Pine Street’, ‘4213413332’);

INSERT INTO Branches
VALUES (‘Parkfront’, ‘Vancouver’, ‘233 Red Way’, ‘6472331239’);

Garages:

INSERT INTO Garages
VALUES (‘Fleetwood’, ‘Surrey'', 34);

INSERT INTO Garages
VALUES (‘Newton’, ‘Surrey’, 12);

INSERT INTO Garages
VALUES (‘Dunbar’, ‘Vancouver’, 8);

INSERT INTO Garages
VALUES (‘Ladner’, ‘Delta’, 30);

INSERT INTO Garages
VALUES (‘Downtown’, ‘Vancouver’, 15);

Offices:

INSERT INTO Offices
VALUES (‘Westwood’, ‘Surrey'', ‘1000-1800’);

INSERT INTO Offices
VALUES (‘East’, ‘Surrey’, ‘1000-1800’);

INSERT INTO Offices
VALUES (‘Ridaeu’, ‘Vancouver’, ‘0900-1900’);

INSERT INTO Offices
VALUES (‘Cader’, ‘Delta’, ‘1000-2000’);

INSERT INTO Offices
VALUES (‘Parkfront’, ‘Vancouver’, ‘0600-1800’);
TrucksParkedAt1:

INSERT INTO TrucksParkedAt1
VALUES (‘H2300’, 20, 100)

INSERT INTO TrucksParkedAt1
VALUES (‘H2400’, 30, 100)

INSERT INTO TrucksParkedAt1

VALUES (‘H1300’, 60, 150)

INSERT INTO TrucksParkedAt1
VALUES (‘H3300’, 53, 100)

INSERT INTO TrucksParkedAt1
VALUES (‘H6300’, 100, 120)

TrucksParkedAt2:

INSERT INTO TrucksParkedAt2
VALUES (‘TRU100’, ‘H2300’, ‘Driving’, ‘2003’, ‘Fleetwood’, ‘Surrey’)

INSERT INTO TrucksParkedAt2
VALUES (‘TRU101’, ‘H2300’, ‘Parked’, ‘2003’, ‘Newton’, ‘Surrey’)

INSERT INTO TrucksParkedAt2
VALUES (‘TRU102’, ‘H1300’, ‘Driving’, ‘2013’, ‘Ladner’, ‘Delta’)

INSERT INTO TrucksParkedAt2
VALUES (‘TRU103’, ‘H3300’, ‘Driving’, ‘2015’, ‘Fleetwood’, ‘Surrey’)

INSERT INTO TrucksParkedAt2
VALUES (‘TRU104’, ‘H6300’, ‘Driving’, ‘2007’, ‘Dubar’, ‘Vancouver’)

Routes1:

INSERT INTO Routes1
VALUES (45, 60)

INSERT INTO Routes1
VALUES (23, 30)

INSERT INTO Routes1
VALUES (55, 70)

INSERT INTO Routes1
VALUES (145, 160)

INSERT INTO Routes1
VALUES (22, 25)

Routes2:

INSERT INTO Routes2
VALUES (‘49.1899541,-122.9577522’, ‘49.1424541,-123.9577532’, 45)

INSERT INTO Routes2
VALUES (‘49.1815697,-122.9622881’, ‘49.1889224,-122.7792763’, 23)

INSERT INTO Routes2
VALUES (‘49.2261325,-122.9578996’, ‘49.1815697,-122.9622881’, 55)

INSERT INTO Routes2
VALUES (‘49.1424541,-123.9577532, ‘49.2261325,-122.9578996’, 145)

INSERT INTO Routes2
VALUES (‘49.2378632,-123.1463217’, ‘49.1424541,-123.9577532’, 22)

OrdersUsedPaid:

INSERT INTO OrdersUsedPaid
VALUES ('ORDER0001', DATE '2024-12-01', 'John', 'Active', 'Fleetwood', 'Newton', 'PAYMENT001');

INSERT INTO OrdersUsedPaid
VALUES ('ORDER0002', DATE '2024-12-02', 'Emily', 'Active', 'Dunbar', 'Surrey', 'PAYMENT002');

INSERT INTO OrdersUsedPaid
VALUES ('ORDER0003', DATE '2024-12-05', 'Alan', 'Pending', 'Ladner', 'Delta', 'PAYMENT003');

INSERT INTO OrdersUsedPaid
VALUES ('ORDER0004', DATE '2024-12-10', 'Nina', 'Shipped', 'Downtown', 'Vancouver', 'PAYMENT004');

INSERT INTO OrdersUsedPaid
VALUES ('ORDER0005', DATE '2024-12-11', 'Laura', 'Canceled','Newton', 'Fleetwood', 'PAYMENT005');

Handles:

```
INSERT INTO Handles
VALUES ('ORDER0001', 'EMP50000');

INSERT INTO Handles
VALUES ('ORDER0002', 'EMP50001');

INSERT INTO Handles
VALUES ('ORDER0003', 'EMP50002');

INSERT INTO Handles
VALUES ('ORDER0004', 'EMP50003');

INSERT INTO Handles
VALUES ('ORDER0005', 'EMP50004');
```

TransportedGoods:

```
INSERT INTO TransportedGoods
VALUES (1000, 'Walmart',  'Fragile',   'ORDER0001');

INSERT INTO TransportedGoods
VALUES ( 500, 'Target',  'Refriger',  'ORDER0002');

INSERT INTO TransportedGoods
VALUES (2000, 'Shoppers',  'DryGoods',  'ORDER0003');

INSERT INTO TransportedGoods
VALUES ( 750, 'Canucks',   'Fragile',   'ORDER0004');

INSERT INTO TransportedGoods
VALUES (1200, 'TwelveWest', 'DryGoods',  'ORDER0005');
```

Payments1:

```
INSERT INTO Payments1
VALUES ( 'Delivery tips 20', 20);

INSERT INTO Payments1
VALUES ( 'Refund 5', -5);

INSERT INTO Payments1
VALUES ( 'Bonus Salary 50', 50);
```

```
INSERT INTO Payments1
VALUES ( 'Damage  100', 100);

INSERT INTO Payments1
VALUES ( 'Repair tips 10', 10);
```

Payments2:

```
INSERT INTO Payments2
VALUES ('PAYMENT001', 'Delivery tips 20', 'RecvrA', 'CustA', 300,  'ORDER0001',
'MAINTEN001');

INSERT INTO Payments2
VALUES ('PAYMENT002', 'Refund 5',   ‘CustA’ , ‘RecvrA’, 150, 'ORDER0002', 'MAINTEN002');

INSERT INTO Payments2
VALUES ('PAYMENT003', 'Bonus Salary 50',    ‘Admin’',    'EMP00000’',     500,  50,
'ORDER0003', 'MAINTEN003');

INSERT INTO Payments2
VALUES ('PAYMENT004', null,   'RecvrD', 'CustD',  700, 'ORDER0004',  'MAINTEN004');

INSERT INTO Payments2
VALUES ('PAYMENT005', 'Repair tips 10',   'RecvrE', 'CustE',   250,  'ORDER0005',
'MAINTEN005');
```

MaintenancesCost:

```
INSERT INTO MaintenancesCost
VALUES ('MAINTEN001', 'OilChg',   'MrLube',  'PAYMENT001');

INSERT INTO MaintenancesCost
VALUES ('MAINTEN002', 'Tires',    'AutoZone', 'PAYMENT002');

INSERT INTO MaintenancesCost
VALUES ('MAINTEN003', 'Engine',   'MechanicShopA',    'PAYMENT003');
```

```
INSERT INTO MaintenancesCost
VALUES ('MAINTEN004', 'Refurb',   'Repair Garage Co',   'PAYMENT004');

INSERT INTO MaintenancesCost
VALUES ('MAINTEN005', 'Brakes',   'Midas',   'PAYMENT005');
```

Maintained:

```
INSERT INTO Maintained
VALUES (DATE '2024-12-10', 'TRU100', 'MAINTEN001');

INSERT INTO Maintained
VALUES (DATE '2024-12-11', 'TRU101', 'MAINTEN002');

INSERT INTO Maintained
VALUES (DATE '2024-12-12', 'TRU102', 'MAINTEN003');

INSERT INTO Maintained
VALUES (DATE '2024-12-13', 'TRU103', 'MAINTEN004');

INSERT INTO Maintained
VALUES (DATE '2024-12-14', 'TRU104', 'MAINTEN005');
```

9) No AI tools are used.