

**Designing & Implementing a**

**Database System**

Taxi Management System

—

Imaad Fahimuddin

—

# INTRODUCTION

This report consolidates the

design and implementation of a relational database system for managing a taxi fleet. The project includes key components such as database schema design, data implementation, complex queries, and advanced functionalities like stored procedures and triggers.

The database is designed to be a Transportation System Database that handle operations like trip management, driver assignments, maintenance schedules, and payment processing while ensuring data integrity and efficiency. This report summarizes all deliverables from previous assignments, providing insights into the system's functionality and practical applications.



**Assignment 1: Business Rules & ER Diagram**

This assignment established the foundation by defining the organization’s business rules and creating an ER diagram. Key entities and relationships were identified to address operational needs such as trip management, driver assignments, and customer interactions.

**Organizational overview:**

Our organization is a transportation company that primarily operates a fleet of taxis. It offers services to customers who require transportation from one location to another. This database will serve multiple users, each having distinct roles and access levels. This database will help the company manage the operation of taxis throughout the day by effectively finding taxi drivers for customers and having a good trip history by saving that kind of data for future analysis.

**Entities and Attributes:**

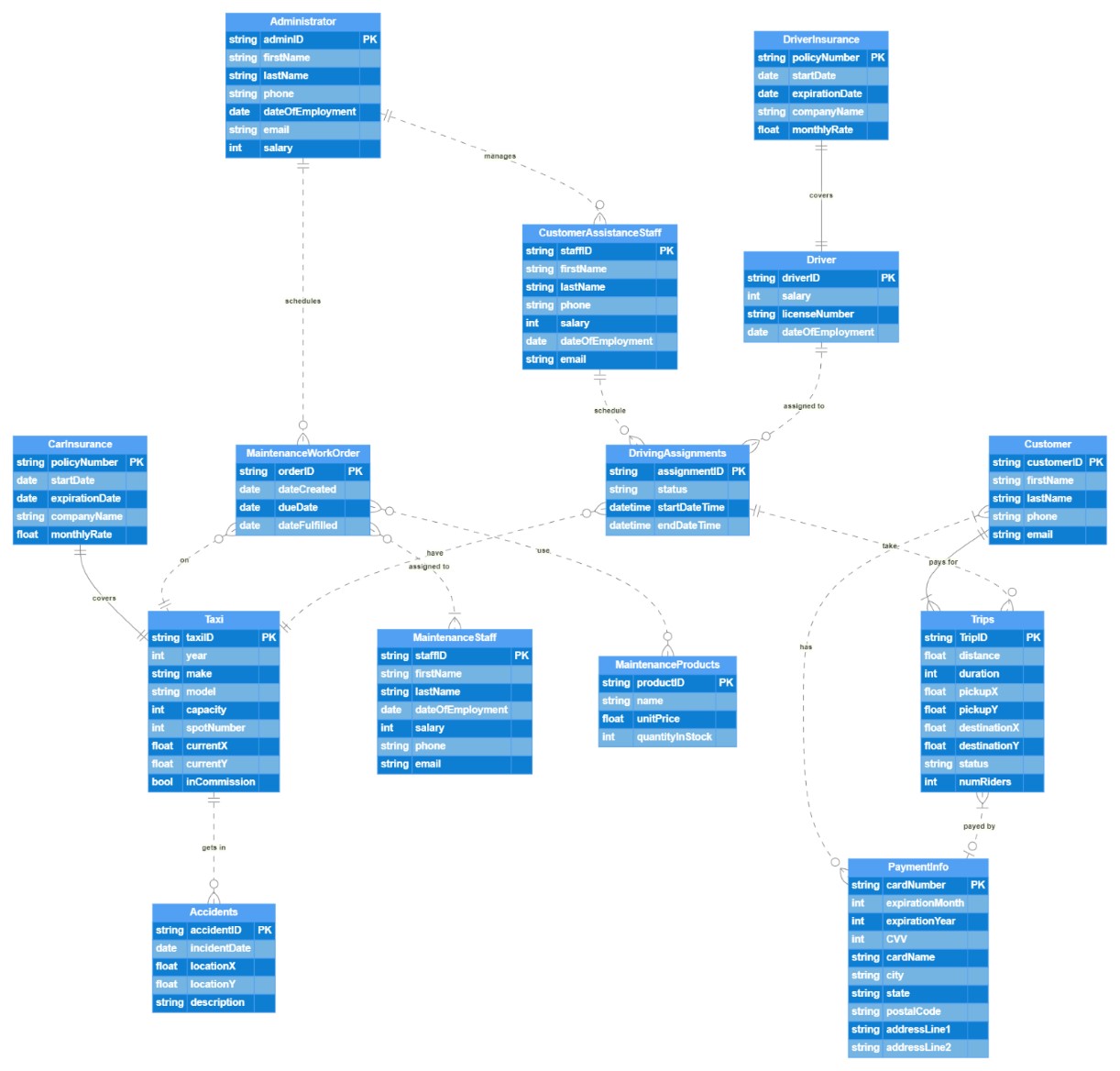
* Taxi– taxiID, year, make, model, capacity, spot number, currentX, currentY, inCommission
* Driver- driverID, salary, licenseNumber, dateOfEmployment
* Driving Assignments – assignmentID, status, startDateTime, endDateTime
* Maintenance Staff – StaffID, first name, last name, date of employment, salary, phone number, email address • Maintenance work order – OrderID, dateCreated, DueDate, DateFulfilled
* Customer – CustomerID, FirstName, LastName, Phone, Email
* Customer Assistance Staff – StaffID, FirstName, LastName, Phone, Salary, DateOfEmployment, Email
* Administrator – AdminID, FirstName, LastName, Phone, Salary, DateOfEmployment, Email
* Maintenance Products – ProductID, product name, unit price, quantity in stock
* Car Insurance – PolicyNumber, StartDate, ExpirationDate, CompanyName, Rate
* Driver Insurance - PolicyNumber, StartDate, ExpirationDate, CompanyName, Rate
* Accidents – AccidentID, incidentDate, locationX, locationY, Description
* Payment info – card number, expirationMonth, expirationYear, CVV, CardName, City, State, Street Address, PostalCode, AddressLine2
* Trips – TripID, distance, duration, pickupX, pickupY, destinationx, destinationY, status, numRiders

**Business Rules:**

* Every trip should have a unique trip ID, including distance, duration, pickup coordinates, destination coordinates, number of riders, and a status to indicate whether the trip is ongoing, reserved, completed, etc.
* Each trip occurs on exactly one driving assignment.
* Each driving assignment may have many or no trips.
* Each driver must have a driver id, salary, and valid driver license, and we also store their date of employment.
* Customer data is stored only after a valid purchase, meaning that all customers have taken and paid for at least one trip. Each trip may have multiple riders, but the trip is only associated with the one rider (customer) who paid for the trip. Customers may take many trips.
* The database stores payment information for any cards that a customer uses. Customers may have multiple or no cards, and each card is associated with at least one customer.
* For payment information, we store the card number, expiration month and year, cvv, the name or company name on the card, and all elements of the billing address.
* Each trip is either paid for in cash (not stored as payment info) or by a single card. One card may be used to pay for many trips.
* Every booking must have a unique ID for the passenger.
* For customers, we store a unique ID, first and last names, as well as email and phone to inform passengers of any delays or cancellations.
* Driving assignments must have exactly one taxi assigned to them.
* A single taxi may have many or no driving assignments.
* For driving assignments, we store a unique ID, the status of the assignment (pending, completed, driver no-show, etc.), and the start and end timestamps.
* For taxis, we store the year, make, model, capacity, storage spot number, current location coordinates, and whether the car is in commission.
* A taxi can have many or no maintenance work orders, but a specific work order can be assigned just to one car at a time.
* Each maintenance work order may use many or no maintenance products.
* Each maintenance product may or may not be used on many work orders.
* For maintenance products, we store a unique product ID, the name of the product, the unit price, and the quantity we have in stock.
* Every taxi car must have one insurance policy which covers exactly one car.
* For the car insurance, we store the policy number, the coverage dates, the name of the insurance company, and the monthly rate of coverage.

Every driver must have one insurance policy which covers only that driver.

* For the driver’s insurance, we store the policy number, the dates of coverage, the name of the insurance company, and the monthly rate of coverage.
* Passengers need to provide contact information.
* Administrators schedule many or no maintenance work orders.
* Each work order is scheduled by a single administrator.
* For maintenance work orders, we store the order ID, the date it was created, the due date, and the date it was fulfilled.
* Administrators manage many or no customer assistance staff (CAS) members.
* CAS members are managed by exactly one administrator.
* For administrators, we store the unique admin ID, first and last names, a phone number, and email, their salary, and their date of employment.
* Customer assistance staff schedule many or no driving assignments.
* For CAS members, we store a unique ID, first and last names, a phone number, an email address, their date of employment, and their salary.
* Each driving assignment is scheduled by one CAS member.
* Maintenance staff work on many or no maintenance work orders.
* Each maintenance work order is worked on by at least one maintenance staff member.
* For maintenance staff, we store a unique staff ID, first and last names, their date of employment, their salary, a phone number and an email address.
* Taxis may be in many or no accidents.
* Each accident is associated with a single taxi.
* For accidents, we store a unique accident ID, the incident date, the location coordinates, and a short description of the incident.





**Assignment 2:**

**Normalization, Data**

**Dictionary and DDL Statements**

The focus was on designing a normalized schema to ensure data integrity and reduce redundancy. The database was normalized to 3NF, capturing all essential attributes while maintaining a structured and scalable schema.

Administrator(adminID : string, firstName : string, lastName : string, phone : string, dateofEmployment : date, email :

string, salary : int)



This table is in 3NF because there are no partial or transitive dependencies.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE NAME** | **Attribute**  **Name** | **Contents** | **Type** | **Format** | **Range** | **Requ**  **ired** | **PK or FK** | **FK**  **REFERENCED**  **TABLE** |
| Administrator | adminID | Administrato r ID | VARCHAR | XXXXXXXXXX | NA | Y | PK | NA |
|  | firstName | First name | VARCHAR | XXXXXXXXXX | NA | Y |  |  |
|  | lastName | Last name | VARCHAR | XXXXXXXXXX | NA | Y |  |  |
|  | phone | Phone number | VARCHAR | XXXXXXXXXX | NA | Y |  |  |
|  | dateOfEmploy ment | Date of employment | DATE | DD-MM-YYYY | NA | Y |  |  |
|  | email | Email address | VARCHAR | XXXXXXXXXX | NA | Y |  |  |
|  | salary | Salary | INT | ###### | 0-  999999 | Y |  |  |

CREATE TABLE Administrator ( adminID VARCHAR(10) PRIMARY KEY, firstName VARCHAR(50) NOT NULL, lastName VARCHAR(50) NOT NULL, phone VARCHAR(15) NOT NULL, dateOfEmployment DATE NOT NULL, email VARCHAR(100) NOT NULL DEFAULT 'not\_provided@example.com', salary INT NOT NULL CHECK (salary > 0),

-- Constraints

CHECK (LENGTH(phone) = 10) -- Phone number must have exactly 10 digits

); CustomerAssistanceStaff(staffID : string, firstName : string, lastName : string, phone : string, dateofEmployment : date, email : string, salary : int, adminID : string)

FK: adminID(Administrator, not Null)



This table is in 3NF because there are no partial or transitive dependencies.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE NAME** | **Attribute**  **Name** | **Contents** | **Type** | **Format** | **Range** | **Requir ed** | **PK or FK** | **FK**  **REFERENCED**  **TABLE** |
| CustomerAssistanc eStaff | staffID | Staff ID | VARCHAR | XXXXXXXXX  X | NA | Y | PK | NA |
|  | firstName | First name | VARCHAR | XXXXXXXXX  X | NA | Y |  |  |
|  | lastName | Last name | VARCHAR | XXXXXXXXX  X | NA | Y |  |  |
|  | phone | Phone number | VARCHAR | XXXXXXXXX  X | NA | Y |  |  |
|  | dateOfEmploy ment | Date of employment | DATE | DD-MM-YYYY | NA | Y |  |  |
|  | email | Email address | VARCHAR | XXXXXXXXX  X | NA | Y |  |  |
|  | salary | Salary | INT | ###### | 0-999999 | Y |  |  |
|  | adminID | Administrato r ID | VARCHAR | XXXXXXXXX  X | NA | Y | FK | ADMINISTRATO  R |

CREATE TABLE CustomerAssistanceStaff ( staffID VARCHAR(10) PRIMARY KEY, firstName VARCHAR(50) NOT NULL, lastName VARCHAR(50) NOT NULL, phone VARCHAR(15) NOT NULL, dateOfEmployment DATE NOT NULL,

email VARCHAR(100) NOT NULL DEFAULT 'not\_provided@example.com', salary INT NOT NULL CHECK (salary > 0), adminID VARCHAR(10) NOT NULL,

-- Foreign Key Constraint

CONSTRAINT fk\_adminID FOREIGN KEY (adminID) REFERENCES Administrator(adminID),

-- Constraints

CHECK (LENGTH(phone) = 10) -- Phone number must have exactly 10 digits

);

DrivingAssignments( assignmentID : string, status : string, startDateTime : datetime, endDateTime : datetime, staffID :

string, driverID : string, taxiID : string)

FK: staffID(CustomerAssistanceStaff, not Null) driverID(Driver, not Null)

taxiID(Taxi, not Null)



This table is in 3NF because there are no partial or transitive dependencies.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE NAME** | **Attribute**  **Name** | **Contents** | **Type** | **Format** | **Range** | **Required** | **PK or FK** | **FK REFERENCED TABLE** |
| DrivingAssignme  nts | assignmentID | Assignment ID | VARCHAR | XXXXXXXXXX | NA | Y | PK | NA |
|  | status | Assignment status | VARCHAR | XXXXXXXXXX XXXXXX | NA | Y |  |  |
|  | startDateTim  e | Start date and time | DATETIME | DD-MM-YYYY HH | NA | Y |  |  |
|  | endDateTime | End date and time | DATETIME | DD-MM-YYYY HH | NA | Y |  |  |
|  | staffID | Customer assistance staff ID | VARCHAR | XXXXXXXXXX | NA | Y | FK | CUSTOMERASSIST ANCESTAFF |
|  | driverID | Driver ID | VARCHAR | XXXXXXXXXX | NA | Y | FK | DRIVER |
|  | taxiID | Taxi ID | VARCHAR | XXXXXXXXXX | NA | Y | FK | TAXI |

CREATE TABLE DrivingAssignments ( assignmentID VARCHAR(10) PRIMARY KEY, status VARCHAR(20) NOT NULL DEFAULT 'pending', startDateTime DATETIME NOT NULL,

endDateTime DATETIME NOT NULL CHECK (endDateTime > startDateTime), staffID VARCHAR(10) NOT NULL, driverID VARCHAR(10) NOT NULL, taxiID VARCHAR(10) NOT NULL, -- Foreign Key Constraints

CONSTRAINT fk\_staffID FOREIGN KEY (staffID) REFERENCES CustomerAssistanceStaff(staffID),

CONSTRAINT fk\_driverID FOREIGN KEY (driverID) REFERENCES Driver(driverID), CONSTRAINT fk\_taxiID FOREIGN KEY (taxiID) REFERENCES Taxi(taxiID),

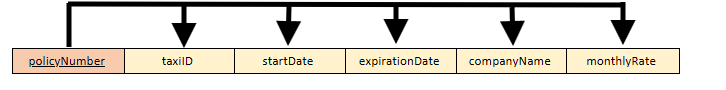
-- Constraints

CHECK (endDateTime > startDateTime) -- End time must be after start time

);

-CarInsurance(policyNumber : string, taxiID : int, startDate : date, expirationDate : date, companyName : string, monthlyRate : int)

FK: taxiID(Taxi, unique, NOT NULL)



This table is in 3NF because there are no partial or transitive dependencies.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE NAME** | **Attribute**  **Name** | **Contents** | **Type** | **Format** | **Range** | **Requir ed** | **PK or FK** | **FK**  **REFERENCED**  **TABLE** |
| CarInsurance | policyNumber | Policy Number | STRING | XXXXXXXXXX | NA | Y | PK | NA |
|  | taxiID | Taxi  Identification | INT | XXXXXXXXXX | NA | Y | FK | Taxi |
|  | startDate | Start Date | DATE | DD-MM-YYYY | NA | Y |  |  |
|  | expirationDate | Expiration Date | DATE | DD-MM-YYYY | NA | Y |  |  |
|  | companyName | Company Name | VARCHAR | XXXXXXXXXX | NA | Y |  |  |
|  | monthlyRate | Monthly Rate | INT | XXXXXXXXXX | NA | Y |  |  |

CREATE TABLE CarInsurance (

policyNumber VARCHAR(50) PRIMARY KEY,

taxiID INT NOT NULL, startDate DATE,

expirationDate DATE, companyName VARCHAR(100),

monthlyRate INT, CONSTRAINT fk\_taxi

FOREIGN KEY (taxiID)

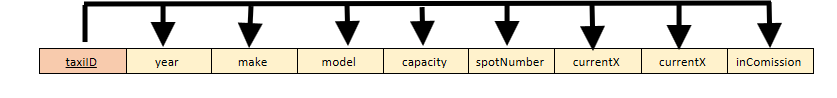
REFERENCES Taxi(taxiID)

ON DELETE CASCADE,

UNIQUE (taxiID)

);

-Taxi(taxiID : int, year : int, make : string, model : string, capacity : int, spotNumber : int, currentX : int, currentY : int, inComission : varchar)



This table is in 3NF because there are no partial or transitive dependencies.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE NAME** | **Attribute**  **Name** | **Contents** | **Type** | **Format** | **Range** | **Required** | **PK or**  **FK** | **FK**  **REFERENCED**  **TABLE** |
| CarInsurance | taxiID | Taxi  Identification | INT | XXXXXXXXXX | NA | Y | PK | NA |
|  | year | Year | INT[4] | XXXX | NA | Y |  |  |
|  | make | Maker | STRING | XXXXXXXXX | NA | Y |  |  |
|  | model | Car Model | STRING | XXXXXXXXX | NA | Y |  |  |
|  | capacity | Car Capacity | INT | XXXXXXXXXX | NA | Y |  |  |
|  | spotNumber | Spot Number | INT | XXXXXXXXXX | NA | Y |  |  |
|  | currentX | X Position in  Map | INT | XXXXXXXXXX | NA | Y |  |  |
|  | currentY | Y Position in  Map | INT | XXXXXXXXXX | NA | Y |  |  |
|  | inComission | Current Taxi  Comision Status | VARCHAR | XXXXXXX | NA | Y |  |  |

CREATE TABLE Taxi (

taxiID INT PRIMARY KEY, year INT,

make VARCHAR(50),

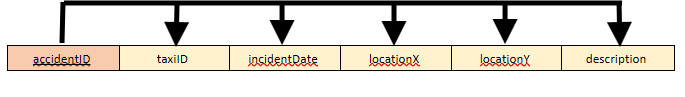
model VARCHAR(50), capacity INT, spotNumber INT, currentX INT,

currentY INT, inComission VARCHAR(10)

);

-Accidents(accidentID, IncidentDate, locationX, locationY, description, taxiID)

FK: taxiID(Taxi, NOT NULL)



This table is in 3NF because there are no partial or transitive dependencies.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **z** | **Attribute Name** | **Contents** | **Type** | **Format** | **Range** | **Requir ed** | **PK or FK** | **FK**  **REFERENCED**  **TABLE** |
| CarInsurance | accidentID | Accident ID | INT | XXXXXXXXXX | NA | Y | PK | NA |
|  | taxiID | Taxi  Identification | INT | XXXXXXXXXX | NA | Y | FK | Taxi |
|  | IncidentDate | Accident Date | DATE | DD-MM-YYYY | NA | Y |  |  |
|  | locationX | X Map  Position of  Accident | INT | XXXXXXXXXX | NA | Y |  |  |
|  | locationY | Y Map Position of Accident | INT | XXXXXXXXXX | NA | Y |  |  |
|  | description | Accident  Description | VARC HAR | XXXXXXXXXX | NA | Y |  |  |

CREATE TABLE Accidents ( accidentID INT PRIMARY KEY,

incidentDate DATE, locationX INT, locationY INT, description VARCHAR(255), taxiID INT NOT NULL, CONSTRAINT fk\_taxi

FOREIGN KEY (taxiID)

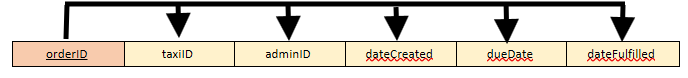
REFERENCES Taxi(taxiID)

ON DELETE CASCADE

);

-MaintenanceWorkOrder(orderID, dateCreated, dueDate, dateFulfilled, taxiID, adminID)

FK: taxiID (Taxi, NOT NULL), adminID(Administrator, NOT NULL)



This table is in 3NF because there are no partial or transitive dependencies.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE NAME** | **Attribute**  **Name** | **Contents** | **Type** | **Format** | **Range** | **Requir ed** | **PK or FK** | **FK**  **REFERENCED**  **TABLE** |
| CarInsurance | orderID | Maintenace Order ID | INT | XXXXXXXXXX | NA | Y | PK | NA |
|  | taxiID | Taxi ID | INT | XXXXXXXXXX | NA | Y | FK | Taxi |
|  | adminID | Admin ID | INT | XXXXXXXXXX | NA | Y | FK | Administrator |
|  | dateCreated | Date Created | DATE | DD-MM-YYYY | NA | Y |  |  |
|  | dueDate | Due Date | DATE | DD-MM-YYYY | NA | Y |  |  |
|  | dateFulfilled | Date Order  Fulfilled | DATE | DD-MM-YYYY | NA | Y |  |  |

CREATE TABLE MaintenanceWorkOrder (

orderID INT PRIMARY KEY,

dateCreated DATE, dueDate DATE, dateFulfilled DATE, taxiID INT NOT NULL, adminID INT NOT NULL, CONSTRAINT fk\_taxi

FOREIGN KEY (taxiID)

REFERENCES Taxi(taxiID)

ON DELETE CASCADE,

CONSTRAINT fk\_admin

FOREIGN KEY (adminID)

REFERENCES Administrator(adminID)

ON DELETE SET NULL

);

Driver (driverID:string, salary:int, licenseNumber:string, dateOfEmployment: date, policyNumber: string)

FK: policyNumber( DriverInsurance, Unique, Not NULL)



This table is in 3NF because there are no partial or transitive dependencies.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE NAME** | **Attribute**  **Name** | **Contents** | **Type** | **Format** | **Range** | **Required** | **PK or FK** | **FK REFERENCED TABLE** |
| Driver | driverID | Driver ID | STRING | XXXXXXXXXX | NA | Y | PK | NA |
|  | salary | Salary | VARCHAR | ###### | 0-  999999 | Y |  |  |
|  | licenseNumb  er | License Number | VARCHAR | XXXXXXXXXX | NA | Y |  |  |
|  | dateOfEmplo yment | Date Of  Employmen  t | DATETIME | DD-MM-YYYY | NA | Y |  |  |
|  | policyNumbe  r | Policy  Number | VARCHAR | XXXXXXXXXX | NA | Y | FK | DriverInsurance |

CREATE TABLE Driver( driverID STRING(10) PRIMARY KEY, salary INT NOT NULL CHECK (salary > 0), dateOfEmployment DATETIME NOT NULL, licenseNumber VARCHAR(10) NOT NULL,

policyNumber VARCHAR(10) NOT NULL, -- Foreign Key Constraints

CONSTRAINT fk\_policyNumber FOREIGN KEY (policyNumber) REFERENCES DriverInsurance( licenseNumber STRING(10) NOT NULL,)

DriverInsurance ( policyNumber: string, startDate: date, expirationDate: date, companyName: string, monthlyRate: int)



This table is in 3NF because there are no partial or transitive dependencies.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE NAME** | **Attribute**  **Name** | **Contents** | **Type** | **Format** | **Range** | **Required** | **PK or FK** | **FK**  **REFERENCED**  **TABLE** |
| DriverInsurance | policyNumber | Policy  Number | VARCHAR | XXXXXXXXXX | NA | Y | PK | NA |
|  | startDate | Start Date | VARCHAR | XXXXXXXXXX XXXXXX | NA | Y |  |  |
|  | expirationDat  e | Expiration Date | DATETIME | DD-MM-YYYY | NA | Y |  |  |
|  | companyNam  e | Company Name | VARCHAR | XXXXXXXXXX | NA | Y |  |  |
|  | monthlyRate | Cost per month | INT | ###### | 0-999999 | Y |  |  |

CREATE TABLE DriverInsurance( policyNumber VARCHAR(10) PRIMARY KEY, startDate DATETIME NOT NULL,

expirationDate DATETIME NOT NULL CHECK (expirationDate> startDate), companyName VARCHAR(10) NOT NULL,

monthlyRate VARCHAR(10) NOT NULL,

-- Constraints

CHECK (expirationDate> startDate) -- Exp time must be after start time

);

Trip (tripID:string, distance:string, duration:date, pickupX:string, pickupY:string, destinationX:string, destinationY:string, status:string, numRiders:string, customerID:string)

FK: customerID ( Customer, Unique, not NULL)



This table is in 3NF because there are no partial or transitive dependencies.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE NAME** | **Attribute**  **Name** | **Contents** | **Type** | **Format** | **Range** | **Require**  **d** | **PK or FK** | **FK**  **REFERENCED**  **TABLE** |
| TRIP | tripID | Trip ID | VARCHAR | XXXXXXXXXX | NA | Y | PK | NA |
|  | distance | distance | VARCHAR | XXXXXXXXXX | NA | Y |  |  |
|  | duration | duration | DATE | XX:XX:XX | 00:00:00-  23:59:59 | Y |  |  |
|  | pickupX | Pickup X | VARCHAR | XXXXXXXXXX | NA | Y |  |  |
|  | pickupY | Pickup Y | VARCHAR | XXXXXXXXXX | NA | Y |  |  |
|  | destinationX | Destination X | VARCHAR | XXXXXXXXXX | NA | Y |  |  |
|  | destinationY | Destination Y | VARCHAR | XXXXXXXXXX | NA | Y |  |  |
|  | status | Trip status | VARCHAR | XXXXXXXXXX | NA | Y |  |  |
|  | numRiders | Num Riders | VARCHAR | XXXXXXXXXX | NA | Y |  |  |
|  | customerID | Customer ID | VARCHAR | XXXXXXXXXX | NA | Y | FK | CUSTOMER |

CREATE TABLE Trip( tripID VARCHAR (10) PRIMARY KEY, distance VARCHAR(20) NOT NULL, duration VARCHAR(10) NOT NULL, pickupX VARCHAR(10) NOT NULL, pickupY VARCHAR(10) NOT NULL, destinationX VARCHAR(10) NOT NULL, destinationY VARCHAR(10) NOT NULL, status VARCHAR(10) NOT NULL, numRiders VARCHAR(10) NOT NULL, customerID VARCHAR(10) NOT NULL,

-- Foreign Key Constraints

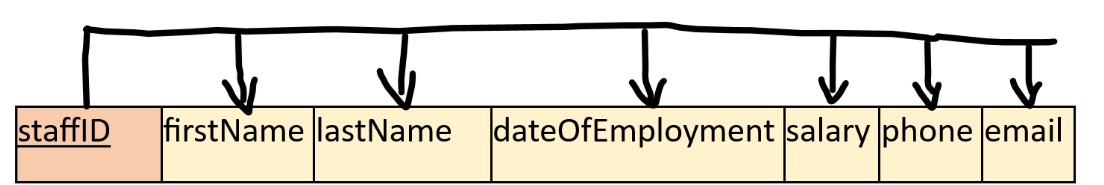
CONSTRAINT fk\_customerID FOREIGN KEY (customerID) REFERENCES Customer(customerID),

-- Constraints

CHECK (duration BETWEEN ‘00:00:00' AND ‘23:59:59’) -- Only accept valid times within the 24-hour-period

);

-MaintenanceStaff(staffID, firstName, lastName, dateOfEmployment, salary, phone, email)



This is in 3nf as there are no transitive and partial dependencies.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE NAME** | **Attribute**  **Name** | **Contents** | **Type** | **Format** | **Range** | **Requir ed** | **PK or FK** | **FK**  **REFERENCED**  **TABLE** |
| TRIP | staffID | Staff ID | INT | #### | NA | Y | PK,FK | maintenanceStaff |
|  | firstName | First name | VARCHAR | XXXXXXX  XXX | NA | Y |  |  |
|  | lastName | Last Name | VARCHAR | XXXX | N/A | Y |  |  |
|  | DateOfEmploym  enrt | DOE | DATE | ##-##-  #### | NA | Y |  |  |
|  | salary | salary | FLOAT | #######.#  # | NA | Y |  |  |
|  | phone | phone | VARCHAR | XXX-XXXXXXX | NA | Y |  |  |
|  | email | email | VARCHAR | XXXXXXX  XXX | NA | Y |  |  |

CREATE TABLE MaintenanceStaff ( staffID INT PRIMARY KEY, firstName VARCHAR(50), lastName VARCHAR(50),

dateOfEmployment DATE,

salary DECIMAL(10, 2), phone VARCHAR(15), email VARCHAR(100),

CONSTRAINT chk\_salary CHECK (salary > 0), -- Salary must be positive

CONSTRAINT chk\_phone CHECK (phone LIKE '[0-9]%') -- Phone format validation

);

MaintenenaceProducts(productID, name, unitPrice, quantityInStock)

product

ID

p

roductName

unitPrice

quantityInSock

This is in 3nf as there are no transitive and partial dependencies.

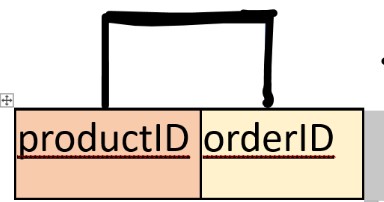
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE NAME** | **Attribute**  **Name** | **Contents** | **Type** | **Format** | **Range** | **Requir ed** | **PK or FK** | **FK**  **REFERENCED**  **TABLE** |
| MaintenancePro ducts | productID | Product ID | VARCHAR | XXXXXXX  XXX | NA | Y | PK | N/A |
|  | productName | Product  Name | VARCHAR | XXXXXXX  XXX | NA | Y |  |  |
|  | unitPrice | Unit Price | DECIMAL | XXXX.XX | N/A | Y |  |  |
|  | quantityInStock | Quantity in Stock | INT | #### | NA | Y |  |  |

CREATE TABLE MaintenanceProducts ( productID VARCHAR(10) PRIMARY KEY, productName VARCHAR(100) NOT NULL, unitPrice DECIMAL(10, 2) NOT NULL CHECK (unitPrice > 0), quantityInStock INT NOT NULL CHECK (quantityInStock >= 0)

);

use(productID, orderID)

Foreign key: productID(MaintenenaceProducts, **not NULL),** orderID(MaintenanceWorkOrder, **not NULL)** Composite Primary keys. OrderID should be underlined.



This is in 3nf as there are no partial dependencies and transitive dependencies.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE NAME** | **Attribute**  **Name** | **Contents** | **Type** | **Format** | **Range** | **Requi red** | **PK or FK** | **FK**  **REFERENCE**  **D TABLE** |
| Use | productID | product ID | INT | #### | NA | Y | PK, FK | maintenenance Product |
|  | orderID | Order ID | INT | #### | NA | Y | PJ,  FK | mainteneanceW orkOrder |

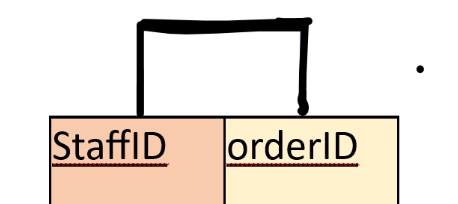
CREATE TABLE Use ( productID INT, orderID INT,

PRIMARY KEY (productID, orderID),

FOREIGN KEY (productID) REFERENCES MaintenanceProducts(productID), FOREIGN KEY (orderID) REFERENCES MaintenanceWorkOrder(orderID)

);

AssignedTo(staffID: INT, orderID: INT)



This is in 3nf as there are no partial dependencies and transitive dependencies.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE NAME** | **Attribute**  **Name** | **Contents** | **Type** | **Format** | **Range** | **Requi red** | **PK or FK** | **FK**  **REFERENCE**  **D TABLE** |
| AssignedTo | staffID | staf ID | INT | #### | NA | Y | PK, FK | maintenenaceSt  aff |
|  | orderID | Order ID | INT | #### | NA | Y | PK,  FK | maintenanceWo rkOrder |

CREATE TABLE AssignedTo (

staffID INT, orderID INT,

PRIMARY KEY (staffID, orderID),

FOREIGN KEY (staffID) REFERENCES MaintenanceStaff(staffID),

FOREIGN KEY (orderID) REFERENCES MaintenanceWorkOrder(orderID)

);

PaymentInfo(CardNumber: string, expirationMonth: int, expirationYear: int, cvv: int, cardName: string, postalCode:

string, addressLine1: string, addressLine2: string)

PK(CardNumber)

FK(postalCode, not Null)



This table is in 3NF because there are no partial or transitive dependencies. Note that city and state were moved to a separate table (PostalCode) to remove transitive dependencies.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE NAME** | **Attribute**  **Name** | **Contents** | **Type** | **Format** | **Range** | **Requir ed** | **PK or FK** | **FK**  **REFERENCED**  **TABLE** |
| PaymentInfo | CardNumber | Number on the card | CHAR(20) | ########  ######## | NA | Y | PK | NA |
|  | expirationMo nth | Month the card expires | INTEGER | ## | [1-12] | Y |  |  |
|  | expirationYea  r | Year the card expires | INTEGER | #### | [1950,  2100] | Y |  |  |
|  | cvv | Security code | INTEGER | #### | 0-9999 | Y |  |  |
|  | cardName | Name on card | VARCHAR(  60) | N/A | N/A | Y |  |  |
|  | postalCode | Billing postal code | CHAR(10) | #####-  #### | N/A | Y | FK | PostalCode |
|  | addressLine1 | Billing  address line  1 | VARCHAR(1  00) | N/A | N/A | Y |  |  |
|  | addressLine2 | Billing address line 2 | VARCHAR(1  00) | N/A | N/A | N |  |  |

CREATE TABLE PaymentInfo(

CardNumber CHAR(20) NOT NULL,

ExpirationMonth INTEGER CHECK(ExpirationMonth >= 1 AND ExpirationMonth <= 12) NOT NULL, ExpirationYear INTEGER CHECK(Expirationyear >=1950 AND ExpirationYear <= 2100) NOT NULL, cvv INTEGER CHECK(cvv >= 0 AND cvv <=9999) NOT NULL,

CardName VARCHAR(60) NOT NULL, PostalCode CHAR(10) NOT NULL, addressLine1 VARCHAR(100) NOT NULL,

AddressLine2 VARCHAR(100),

PRIMARY KEY (CardNumber),

FOREIGN KEY (postalCode) REFERENCES PostalCode(postalCode)

);

PostalCode(PostalCode: string, city: string, state: string)

PK(CustomerID)



PK(PostalCode)



This table is in 3NF because there are no partial or transitive dependencies. (Cities do not uniquely identify states.)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE NAME** | **Attribute**  **Name** | **Contents** | **Type** | **Format** | **Range** | **Requi red** | **PK or FK** | **FK**  **REFERENCED**  **TABLE** |
| PostalCode | postalCode | Billing postal code | CHAR(10) | #####-  #### | N/A | Y | PK |  |
|  | city | Billing City | VARCHAR(  60) | N/A | N/A | Y |  |  |
|  | state | Billing State | CHAR(2) | XX | N/A | Y |  |  |
|  |  |  |  |  |  |  |  |  |

CREATE TABLE PostalCode( PostalCode CHAR(10) NOT NULL, city VARCHAR(60) NOT NULL, state CHAR(2) NOT NULL, PRIMARY KEY (postalCode)

);

Customer(CustomerID: string, firstName: string, lastName: string, phone: string, email: string)

This table is in 3NF because there are no partial or transitive dependencies.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE NAME** | **Attribute**  **Name** | **Contents** | **Type** | **Format** | **Range** | **Requir ed** | **PK or FK** | **FK**  **REFERENCED**  **TABLE** |
| Customer | CustomerID | Unique  identifier for each customer | CHAR(10) | xxxxxxxxxx | N/A | Y | PK |  |
|  | firstName | First name | VARCHAR(50  ) | N/A | N/A | Y |  |  |
|  | lastName | Last name | VARCHAR(60  ) | N/A | N/A | Y |  |  |
|  | phone | Customer phone number | VARCHAR(17) | (###)###-    ###-#### | N/A | Y |  |  |
|  | email | Customer email | Varchar 50 | N/A | N/A | N |  |  |

CREATE TABLE Customer( customerID CHAR(10) NOT NULL, firstName VARCHAR(50) NOT NULL, lastName VARCHAR(50) NOT NULL, phone VARCHAR(17) NOT NULL, email VARCHAR(50) NOT NULL,

PRIMARY KEY(customerID)

);

CustomerPaymentInfoLink(customerID: string, cardNumber: string)

PK(customerID, cardNumber) FK(customerID, cardNumber)



This table is in 3NF because there are no partial or transitive dependencies.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TABLE NAME** | **Attribute**  **Name** | **Contents** | **Type** | **Format** | **Range** | **Requir ed** | **PK or FK** | **FK**  **REFERENCED**  **TABLE** |
| CustomerPayment  InfoLink | customerID | Unique  identifier for each customer | CHAR(10) | xxxxxxxxxx | N/A | Y | PK and FK | Customer |
|  | cardNumber | Number on the card | CHAR(20) | ######## ######## | NA | Y | PK and FK | PaymentInfo |

CREATE TABLE CustomerPaymentInfoLink( customerID CHAR(10) NOT NULL, cardNumber CHAR(20) NOT NULL, PRIMARY KEY (customerID, cardNumber),

FORIEGN KEY (customerID) REFERENCES Customer(customerID),

FORIEGN KEY (cardNumber) REERENCES PaymentInfo(cardNumber)

);



**Assignment 3: Data, Queries & Updates**

Data scripts were created to insert initial records into the database. Additionally, update scripts were designed to simulate real-world data modifications, such as adjusting driver salaries, updating maintenance work orders, and modifying trip details.

## CREATE.txt

CREATE TABLE Administrator ( adminID VARCHAR(10) PRIMARY KEY, firstName VARCHAR(50) NOT NULL, lastName VARCHAR(50) NOT NULL, phone VARCHAR(15) NOT NULL, dateOfEmployment DATE NOT NULL,

email VARCHAR(100) DEFAULT ('not\_provided@example.com'),

salary INT NOT NULL CHECK (salary > 0),

CHECK (LENGTH(phone) = 10) -- Phone number must have exactly 10 digits

);

CREATE TABLE CustomerAssistanceStaff ( staffID VARCHAR(10) PRIMARY KEY, firstName VARCHAR(50) NOT NULL, lastName VARCHAR(50) NOT NULL, phone VARCHAR(15) NOT NULL, dateOfEmployment DATE NOT NULL,

email VARCHAR(100) DEFAULT ('not\_provided@example.com'),

salary INT NOT NULL CHECK (salary > 0), adminID VARCHAR(10) NOT NULL,

CONSTRAINT fk\_adminID FOREIGN KEY (adminID) REFERENCES Administrator(adminID), CHECK (LENGTH(phone) = 10)

);

CREATE TABLE DriverInsurance ( policyNumber VARCHAR(10) PRIMARY KEY,

startDate DATE NOT NULL, expirationDate DATE NOT NULL, companyName VARCHAR(10) NOT NULL,

monthlyRate INT NOT NULL,

-- Table-level constraint

CHECK (expirationDate > startDate) -- Expiration time must be after start time

);

CREATE TABLE Driver ( driverID VARCHAR(10) PRIMARY KEY, salary INT NOT NULL CHECK (salary > 0), dateOfEmployment DATE NOT NULL, licenseNumber VARCHAR(10) NOT NULL , policyNumber VARCHAR(10) NOT NULL, -- Foreign Key Constraints

CONSTRAINT fk\_policyNumber FOREIGN KEY (policyNumber) REFERENCES DriverInsurance(policyNumber)

);

CREATE TABLE Customer( customerID VARCHAR(10) NOT NULL, firstName VARCHAR(50) NOT NULL, lastName VARCHAR(50) NOT NULL, phone VARCHAR(17) NOT NULL, email VARCHAR(50) NOT NULL,

PRIMARY KEY(customerID)

);

CREATE TABLE PostalCode( PostalCode CHAR(10) NOT NULL, city VARCHAR(60) NOT NULL, state CHAR(2) NOT NULL, PRIMARY KEY (postalCode)

);

CREATE TABLE PaymentInfo(

CardNumber CHAR(20) NOT NULL,

ExpirationMonth INTEGER CHECK(ExpirationMonth >= 1 AND ExpirationMonth <= 12) NOT NULL, ExpirationYear INTEGER CHECK(Expirationyear >=1950 AND ExpirationYear <= 2100) NOT NULL, cvv INTEGER CHECK(cvv >= 0 AND cvv <=9999) NOT NULL,

CardName VARCHAR(60) NOT NULL, PostalCode CHAR(10) NOT NULL, addressLine1 VARCHAR(100) NOT NULL,

AddressLine2 VARCHAR(100),

PRIMARY KEY (CardNumber),

CONSTRAINT fk\_postalCode FOREIGN KEY (postalCode) REFERENCES PostalCode(postalCode)

);

CREATE TABLE Trip( tripID VARCHAR (10) PRIMARY KEY, distance DECIMAL(5, 2) NOT NULL, duration VARCHAR(10) NOT NULL, pickupX VARCHAR(10) NOT NULL, pickupY VARCHAR(10) NOT NULL, destinationX VARCHAR(10) NOT NULL, destinationY VARCHAR(10) NOT NULL, status VARCHAR(10) NOT NULL, numRiders INT NOT NULL, cardNumber CHAR(20) NOT NULL,

customerID VARCHAR(10) NOT NULL, -- Foreign Key Constraints

CONSTRAINT fk\_customerID FOREIGN KEY (customerID) REFERENCES Customer(customerID),

CONSTRAINT fk\_cardNumber2 FOREIGN KEY (cardNumber) REFERENCES PaymentInfo (cardNumber),

-- Constraints

CHECK (duration BETWEEN '00:00:00' AND '23:59:59') -- Only accept valid times within the 24-hour period

);

-- Creating MaintenanceStaff Table CREATE TABLE MaintenanceStaff ( staffID VARCHAR(10) PRIMARY KEY, firstName VARCHAR(50) NOT NULL, lastName VARCHAR(50) NOT NULL, phone VARCHAR(15) NOT NULL, dateOfEmployment DATE NOT NULL,

email VARCHAR(100) DEFAULT ('not\_provided@example.com'), salary DECIMAL(10, 2) NOT NULL CHECK (salary > 0)

);

-- Creating MaintenanceProducts Table CREATE TABLE MaintenanceProducts ( productID VARCHAR(10) PRIMARY KEY, productName VARCHAR(100) NOT NULL, unitPrice DECIMAL(10, 2) NOT NULL CHECK (unitPrice > 0), quantityInStock INT NOT NULL CHECK (quantityInStock >= 0)

);

CREATE TABLE Taxi (

taxiID VARCHAR(10) PRIMARY KEY, year INT,

make VARCHAR(50),

model VARCHAR(50), capacity INT, spotNumber INT, currentX FLOAT, currentY FLOAT,

inComission VARCHAR(10)

);

CREATE TABLE MaintenanceWorkOrder ( orderID VARCHAR(10) PRIMARY KEY,

dateCreated DATE NOT NULL,

dueDate DATE NOT NULL, dateFulfilled DATE, taxiID VARCHAR(10) NOT NULL, adminID VARCHAR(10) NOT NULL,

CONSTRAINT fk\_taxiID\_MWO FOREIGN KEY (taxiID) REFERENCES Taxi(taxiID) ON DELETE CASCADE,

CONSTRAINT fk\_adminID\_MWO FOREIGN KEY (adminID) REFERENCES Administrator(adminID) ON DELETE SET NULL,

CHECK (dueDate > dateCreated)

);

-- Creating Use Table CREATE TABLE Use ( productID VARCHAR(10), orderID VARCHAR(10),

PRIMARY KEY (productID, orderID),

CONSTRAINT fk\_productID FOREIGN KEY (productID) REFERENCES MaintenanceProducts(productID), CONSTRAINT fk\_orderID FOREIGN KEY (orderID) REFERENCES MaintenanceWorkOrder(orderID)

);

-- Creating AssignedTo Table CREATE TABLE AssignedTo ( staffID VARCHAR(10), orderID VARCHAR(10),

PRIMARY KEY (staffID, orderID),

CONSTRAINT fk\_staffID\_link FOREIGN KEY (staffID) REFERENCES MaintenanceStaff(staffID),

CONSTRAINT fk\_orderID\_link FOREIGN KEY (orderID) REFERENCES MaintenanceWorkOrder(orderID)

);

CREATE TABLE CustomerPaymentInfoLink( customerID VARCHAR(10) NOT NULL, cardNumber CHAR(20) NOT NULL, PRIMARY KEY (customerID, cardNumber),

CONSTRAINT fk\_customerID\_link FOREIGN KEY (customerID) REFERENCES Customer(customerID),

CONSTRAINT fk\_cardNumber\_link FOREIGN KEY (cardNumber) REFERENCES PaymentInfo(cardNumber)

);

CREATE TABLE CarInsurance ( policyNumber VARCHAR(10) PRIMARY KEY, taxiID VARCHAR(10) NOT NULL, startDate DATE NOT NULL, expirationDate DATE NOT NULL, companyName VARCHAR(100), monthlyRate FLOAT,

CONSTRAINT fk\_taxiID\_CI FOREIGN KEY (taxiID) REFERENCES Taxi(taxiID) ON DELETE CASCADE, UNIQUE (taxiID),

CHECK( expirationDate >startDate )

);

CREATE TABLE Accidents (

accidentID VARCHAR(10) PRIMARY KEY, incidentDate DATE NOT NULL,

locationX FLOAT, locationY FLOAT, description VARCHAR(255), taxiID VARCHAR(10) NOT NULL,

CONSTRAINT fk\_taxiID\_A FOREIGN KEY (taxiID) REFERENCES Taxi(taxiID) ON DELETE CASCADE

);

CREATE TABLE DrivingAssignments ( assignmentID VARCHAR(10) PRIMARY KEY, status VARCHAR(20) DEFAULT 'pending', startDateTime TIMESTAMP NOT NULL, endDateTime TIMESTAMP NOT NULL, staffID VARCHAR(10) NOT NULL, driverID VARCHAR(10) NOT NULL, taxiID VARCHAR(10) NOT NULL,

CONSTRAINT fk\_staffID FOREIGN KEY (staffID) REFERENCES CustomerAssistanceStaff(staffID),

CONSTRAINT fk\_driverID FOREIGN KEY (driverID) REFERENCES Driver(driverID),

CONSTRAINT fk\_taxiID FOREIGN KEY (taxiID) REFERENCES Taxi(taxiID), CHECK (endDateTime > startDateTime)

);

## INSERT.txt

-- Inserting data into Administrator Table

INSERT INTO Administrator (adminID, firstName, lastName, phone, dateOfEmployment, email, salary)

VALUES ('ADM001', 'John', 'Doe', '1234567890', TO\_DATE('2023-01-10', 'YYYY-MM-DD'), 'johndoe@example.com', 60000);

INSERT INTO Administrator (adminID, firstName, lastName, phone, dateOfEmployment, email, salary)

VALUES ('ADM002', 'Jane', 'Smith', '0987654321', TO\_DATE('2022-05-15', 'YYYY-MM-DD'), 'janesmith@example.com', 75000);

INSERT INTO Administrator (adminID, firstName, lastName, phone, dateOfEmployment, email, salary)

VALUES ('ADM003', 'Alice', 'Johnson', '1029384756', TO\_DATE('2023-02-25', 'YYYY-MM-DD'), 'alicej@example.com', 62000);

INSERT INTO Administrator (adminID, firstName, lastName, phone, dateOfEmployment, email, salary)

VALUES ('ADM004', 'Robert', 'Brown', '5647382910', TO\_DATE('2023-03-15', 'YYYY-MM-DD'), 'robertb@example.com', 68000);

INSERT INTO Administrator (adminID, firstName, lastName, phone, dateOfEmployment, email, salary)

VALUES ('ADM005', 'Emily', 'Davis', '9876543210', TO\_DATE('2022-08-10', 'YYYY-MM-DD'), 'emilyd@example.com', 71000);

INSERT INTO Administrator (adminID, firstName, lastName, phone, dateOfEmployment, email, salary)

VALUES ('ADM006', 'Chris', 'Wilson', '1234509876', TO\_DATE('2021-12-01', 'YYYY-MM-DD'), 'chrisw@example.com', 64000);

INSERT INTO Administrator (adminID, firstName, lastName, phone, dateOfEmployment, email, salary)

VALUES ('ADM007', 'Patricia', 'Miller', '1122334455', TO\_DATE('2020-05-20', 'YYYY-MM-DD'), 'patm@example.com', 63000);

INSERT INTO Administrator (adminID, firstName, lastName, phone, dateOfEmployment, email, salary)

VALUES ('ADM008', 'Linda', 'Taylor', '9988776655', TO\_DATE('2021-09-12', 'YYYY-MM-DD'), 'lindat@example.com', 72000);

INSERT INTO Administrator (adminID, firstName, lastName, phone, dateOfEmployment, email, salary)

VALUES ('ADM009', 'Michael', 'Clark', '5566778899', TO\_DATE('2022-11-18', 'YYYY-MM-DD'), 'michaelc@example.com', 67000);

INSERT INTO Administrator (adminID, firstName, lastName, phone, dateOfEmployment, email, salary)

VALUES ('ADM010', 'Karen', 'Lee', '2233445566', TO\_DATE('2023-06-01', 'YYYY-MM-DD'), 'karenl@example.com', 65000);

-- Inserting data into CustomerAssistanceStaff Table

INSERT INTO CustomerAssistanceStaff (staffID, firstName, lastName, phone, dateOfEmployment, email, salary, adminID)

VALUES ('CAS001', 'Matthew', 'Green', '5551112233', TO\_DATE('2022-04-15', 'YYYY-MM-DD'),

'matthewg@example.com', 45000, 'ADM001');

INSERT INTO CustomerAssistanceStaff (staffID, firstName, lastName, phone, dateOfEmployment, email, salary, adminID)

VALUES ('CAS002', 'Sophia', 'Young', '5552223344', TO\_DATE('2021-12-20', 'YYYY-MM-DD'), 'sophiay@example.com', 46000, 'ADM002');

INSERT INTO CustomerAssistanceStaff (staffID, firstName, lastName, phone, dateOfEmployment, email, salary, adminID)

VALUES ('CAS003', 'Daniel', 'Martinez', '5553334455', TO\_DATE('2023-01-10', 'YYYY-MM-DD'), 'danielm@example.com', 47000, 'ADM003');

INSERT INTO CustomerAssistanceStaff (staffID, firstName, lastName, phone, dateOfEmployment, email, salary, adminID)

VALUES ('CAS004', 'Olivia', 'Hernandez', '5554445566', TO\_DATE('2022-08-01', 'YYYY-MM-DD'), 'oliviah@example.com', 44000, 'ADM004');

INSERT INTO CustomerAssistanceStaff (staffID, firstName, lastName, phone, dateOfEmployment, email, salary, adminID)

VALUES ('CAS005', 'William', 'Lopez', '5555556677', TO\_DATE('2021-09-25', 'YYYY-MM-DD'), 'williaml@example.com', 48000, 'ADM005');

INSERT INTO CustomerAssistanceStaff (staffID, firstName, lastName, phone, dateOfEmployment, email, salary, adminID)

VALUES ('CAS006', 'Emma', 'Gonzalez', '5556667788', TO\_DATE('2022-03-15', 'YYYY-MM-DD'), 'emmag@example.com', 46000, 'ADM006');

INSERT INTO CustomerAssistanceStaff (staffID, firstName, lastName, phone, dateOfEmployment, email, salary, adminID)

VALUES ('CAS007', 'James', 'Perez', '5557778899', TO\_DATE('2023-05-10', 'YYYY-MM-DD'), 'jamesp@example.com', 47000, 'ADM007');

INSERT INTO CustomerAssistanceStaff (staffID, firstName, lastName, phone, dateOfEmployment, email, salary, adminID)

VALUES ('CAS008', 'Isabella', 'Kim', '5558889900', TO\_DATE('2022-07-25', 'YYYY-MM-DD'), 'isabellak@example.com', 45000, 'ADM008');

INSERT INTO CustomerAssistanceStaff (staffID, firstName, lastName, phone, dateOfEmployment, email, salary, adminID)

VALUES ('CAS009', 'Benjamin', 'Rivera', '5559990011', TO\_DATE('2021-06-15', 'YYYY-MM-DD'),

'benjaminr@example.com', 46000, 'ADM009');

INSERT INTO CustomerAssistanceStaff (staffID, firstName, lastName, phone, dateOfEmployment, email, salary, adminID)

VALUES ('CAS010', 'Mia', 'Garcia', '5550001122', TO\_DATE('2023-02-20', 'YYYY-MM-DD'), 'miag@example.com', 49000, 'ADM010');

-- Insert data into DriverInsurance table

INSERT INTO DriverInsurance (policyNumber, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL001', TO\_DATE('2024-01-01', 'YYYY-MM-DD'), TO\_DATE('2025-01-01', 'YYYY-MM-DD'), 'InsureCo', 150.00);

INSERT INTO DriverInsurance (policyNumber, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL002', TO\_DATE('2024-02-15', 'YYYY-MM-DD'), TO\_DATE('2025-02-15', 'YYYY-MM-DD'), 'SafeLife', 120.00);

INSERT INTO DriverInsurance (policyNumber, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL003', TO\_DATE('2024-03-10', 'YYYY-MM-DD'), TO\_DATE('2025-03-10', 'YYYY-MM-DD'), 'ProtectUs', 200.00);

INSERT INTO DriverInsurance (policyNumber, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL004', TO\_DATE('2024-04-05', 'YYYY-MM-DD'), TO\_DATE('2025-04-05', 'YYYY-MM-DD'), 'CoverNow', 110.00);

INSERT INTO DriverInsurance (policyNumber, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL005', TO\_DATE('2024-05-20', 'YYYY-MM-DD'), TO\_DATE('2025-05-20', 'YYYY-MM-DD'), 'ShieldPlus', 130.00);

INSERT INTO DriverInsurance (policyNumber, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL006', TO\_DATE('2024-06-15', 'YYYY-MM-DD'), TO\_DATE('2025-06-15', 'YYYY-MM-DD'), 'SecureIt', 175.00);

INSERT INTO DriverInsurance (policyNumber, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL007', TO\_DATE('2024-07-30', 'YYYY-MM-DD'), TO\_DATE('2025-07-30', 'YYYY-MM-DD'), 'InsureCo', 140.00);

INSERT INTO DriverInsurance (policyNumber, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL008', TO\_DATE('2024-08-25', 'YYYY-MM-DD'), TO\_DATE('2025-08-25', 'YYYY-MM-DD'), 'SafeLife', 160.00);

INSERT INTO DriverInsurance (policyNumber, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL009', TO\_DATE('2024-09-18', 'YYYY-MM-DD'), TO\_DATE('2025-09-18', 'YYYY-MM-DD'), 'ProtectUs', 190.00);

INSERT INTO DriverInsurance (policyNumber, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL010', TO\_DATE('2024-10-22', 'YYYY-MM-DD'), TO\_DATE('2025-10-22', 'YYYY-MM-DD'), 'CoverNow', 155.00);

-- Insert data into Driver table

INSERT INTO Driver (driverID, salary, dateOfEmployment, licenseNumber, policyNumber) VALUES ('DRV001', 45000, TO\_DATE('2020-01-15', 'YYYY-MM-DD'), 'LIC001', 'POL001');

INSERT INTO Driver (driverID, salary, dateOfEmployment, licenseNumber, policyNumber)

VALUES ('DRV002', 52000, TO\_DATE('2019-03-10', 'YYYY-MM-DD'), 'LIC002', 'POL002');

INSERT INTO Driver (driverID, salary, dateOfEmployment, licenseNumber, policyNumber) VALUES ('DRV003', 47000, TO\_DATE('2018-05-20', 'YYYY-MM-DD'), 'LIC003', 'POL003');

INSERT INTO Driver (driverID, salary, dateOfEmployment, licenseNumber, policyNumber)

VALUES ('DRV004', 53000, TO\_DATE('2021-07-30', 'YYYY-MM-DD'), 'LIC004', 'POL004');

INSERT INTO Driver (driverID, salary, dateOfEmployment, licenseNumber, policyNumber) VALUES ('DRV005', 56000, TO\_DATE('2017-09-18', 'YYYY-MM-DD'), 'LIC005', 'POL005');

INSERT INTO Driver (driverID, salary, dateOfEmployment, licenseNumber, policyNumber)

VALUES ('DRV006', 49000, TO\_DATE('2022-02-12', 'YYYY-MM-DD'), 'LIC006', 'POL006');

INSERT INTO Driver (driverID, salary, dateOfEmployment, licenseNumber, policyNumber) VALUES ('DRV007', 58000, TO\_DATE('2021-10-25', 'YYYY-MM-DD'), 'LIC007', 'POL007');

INSERT INTO Driver (driverID, salary, dateOfEmployment, licenseNumber, policyNumber)

VALUES ('DRV008', 46000, TO\_DATE('2019-12-08', 'YYYY-MM-DD'), 'LIC008', 'POL008');

INSERT INTO Driver (driverID, salary, dateOfEmployment, licenseNumber, policyNumber) VALUES ('DRV009', 50000, TO\_DATE('2023-04-05', 'YYYY-MM-DD'), 'LIC009', 'POL009');

INSERT INTO Driver (driverID, salary, dateOfEmployment, licenseNumber, policyNumber) VALUES ('DRV010', 55000, TO\_DATE('2022-08-15', 'YYYY-MM-DD'), 'LIC010', 'POL010');

--Customer inserts

INSERT INTO Customer (customerID, firstName, lastName, phone, email)

VALUES ('0000000001', 'John', 'Doe', '202-555-0123', 'john.doe@example.com');

INSERT INTO Customer (customerID, firstName, lastName, phone, email)

VALUES ('0000000002', 'Jane', 'Smith', '303-555-0456', 'jane.smith@example.com');

INSERT INTO Customer (customerID, firstName, lastName, phone, email)

VALUES ('0000000003', 'Emily', 'Johnson', '415-555-0789', 'emily.johnson@example.com');

INSERT INTO Customer (customerID, firstName, lastName, phone, email)

VALUES ('0000000004', 'Michael', 'Brown', '512-555-1011', 'michael.brown@example.com');

INSERT INTO Customer (customerID, firstName, lastName, phone, email)

VALUES ('0000000005', 'Sarah', 'Davis', '618-555-1213', 'sarah.davis@example.com');

INSERT INTO Customer (customerID, firstName, lastName, phone, email)

VALUES ('0000000006', 'David', 'Martinez', '707-555-1415', 'david.martinez@example.com');

INSERT INTO Customer (customerID, firstName, lastName, phone, email)

VALUES ('0000000007', 'Jessica', 'Garcia', '818-555-1617', 'jessica.garcia@example.com');

INSERT INTO Customer (customerID, firstName, lastName, phone, email)

VALUES ('0000000008', 'Daniel', 'Miller', '919-555-1819', 'daniel.miller@example.com');

INSERT INTO Customer (customerID, firstName, lastName, phone, email)

VALUES ('0000000009', 'Sophia', 'Wilson', '202-555-2021', 'sophia.wilson@example.com');

INSERT INTO Customer (customerID, firstName, lastName, phone, email)

VALUES ('000000000A', 'James', 'Taylor', '303-555-2223', 'james.taylor@example.com');

--PostalCode inserts

INSERT INTO PostalCode (PostalCode,city, state) VALUES ('33618-3224', 'Tampa', 'FL');

INSERT INTO PostalCode (PostalCode, city, state) VALUES ('32801-1120', 'Orlando', 'FL');

INSERT INTO PostalCode (PostalCode, city, state) VALUES ('32803-4512', 'Orlando', 'FL');

INSERT INTO PostalCode (PostalCode, city, state) VALUES ('32819-7843', 'Orlando', 'FL');

INSERT INTO PostalCode (PostalCode, city, state) VALUES ('33301-2204', 'Fort Lauderdale', 'FL');

INSERT INTO PostalCode (PostalCode, city, state) VALUES ('33304-3012', 'Fort Lauderdale', 'FL');

INSERT INTO PostalCode (PostalCode, city, state) VALUES ('33316-5801', 'Fort Lauderdale', 'FL');

INSERT INTO PostalCode (PostalCode, city, state) VALUES ('32601-5304', 'Gainesville', 'FL');

INSERT INTO PostalCode (PostalCode, city, state) VALUES ('32608-1123', 'Gainesville', 'FL');

INSERT INTO PostalCode (PostalCode, city, state) VALUES ('32607-4510', 'Gainesville', 'FL');

INSERT INTO PostalCode (PostalCode, city, state) VALUES ('32801-1234', 'Wk', 'FA'); INSERT INTO PostalCode (PostalCode, city, state) VALUES ('90001-5678', 'Ls', 'CW');

INSERT INTO PostalCode (PostalCode, city, state) VALUES ('10001-9101', 'Nk', 'NK'); INSERT INTO PostalCode (PostalCode, city, state) VALUES ('60601-1122', 'Co', 'IX');

--Payment info inserts

INSERT INTO PaymentInfo (CardNumber, ExpirationMonth, ExpirationYear, cvv, CardName, PostalCode, addressLine1, addressLine2) VALUES ('4532123456789010', 05, 2023, 1234, 'John Doe', '32801-1120', '123 Main St', NULL);

INSERT INTO PaymentInfo (CardNumber, ExpirationMonth, ExpirationYear, cvv, CardName, PostalCode, addressLine1, addressLine2) VALUES ('4012345678901234', 11, 1998, 5678, 'Jane Smith', '33301-2204', '456 Elm St', 'Apt 3B'); INSERT INTO PaymentInfo (CardNumber, ExpirationMonth, ExpirationYear, cvv, CardName, PostalCode, addressLine1, addressLine2) VALUES ('6011234567890123', 01, 2050, 9101, 'Emily Johnson', '33618-3224', '789 Oak St', NULL); INSERT INTO PaymentInfo (CardNumber, ExpirationMonth, ExpirationYear, cvv, CardName, PostalCode, addressLine1, addressLine2) VALUES ('4222222222222222', 07, 2015, 2345, 'Michael Brown', '32803-4512', '321 Pine St', NULL); INSERT INTO PaymentInfo (CardNumber, ExpirationMonth, ExpirationYear, cvv, CardName, PostalCode, addressLine1, addressLine2) VALUES ('378282246310005', 03, 2027, 3456, 'Sarah Davis', '32819-7843', '654 Maple St', 'Suite 200'); INSERT INTO PaymentInfo (CardNumber, ExpirationMonth, ExpirationYear, cvv, CardName, PostalCode, addressLine1, addressLine2) VALUES ('6011123456789010', 09, 2010, 4567, 'David Martinez', '33304-3012', '987 Cedar St', NULL); INSERT INTO PaymentInfo (CardNumber, ExpirationMonth, ExpirationYear, cvv, CardName, PostalCode, addressLine1, addressLine2) VALUES ('5111111111111111', 12, 1995, 6789, 'Jessica Garcia', '32601-5304', '159 Birch St', NULL); INSERT INTO PaymentInfo (CardNumber, ExpirationMonth, ExpirationYear, cvv, CardName, PostalCode, addressLine1, addressLine2) VALUES ('5105101051055100', 02, 2005, 2340, 'Daniel Miller', '32608-1123', '753 Willow St', 'Floor 4'); INSERT INTO PaymentInfo (CardNumber, ExpirationMonth, ExpirationYear, cvv, CardName, PostalCode, addressLine1, addressLine2) VALUES ('4485485485485485', 06, 2024, 5432, 'James Taylor', '32819-7843', '321 Oak Ave', 'Building A'); INSERT INTO PaymentInfo (CardNumber, ExpirationMonth, ExpirationYear, cvv, CardName, PostalCode, addressLine1, addressLine2) VALUES ('4111111111111111', 06, 2024, 2321,'Sophia Wilson', '33618-3224', '321 Grape Run', NULL);

-- Insert data into Trip table

INSERT INTO Trip (tripID, distance, duration, pickupX, pickupY, destinationX, destinationY, status, numRiders, cardNumber, customerID)

VALUES ('TRP001', 12.5, '00:25:00', '40.7128N', '74.0060W', '40.7306N', '73.9352W', 'completed', 2, '4532123456789010','0000000001');

INSERT INTO Trip (tripID, distance, duration, pickupX, pickupY, destinationX, destinationY, status, numRiders, cardNumber, customerID)

VALUES ('TRP002', 5.0, '00:15:00', '40.7306N', '73.9352W', '40.7580N', '73.9855W', 'completed', 1, '4012345678901234', '0000000002');

INSERT INTO Trip (tripID, distance, duration, pickupX, pickupY, destinationX, destinationY, status, numRiders, cardNumber, customerID)

VALUES ('TRP003', 7.3, '00:20:00', '40.7580N', '73.9855W', '40.7291N', '73.9965W', 'cancelled', 3, '6011234567890123','0000000003');

INSERT INTO Trip (tripID, distance, duration, pickupX, pickupY, destinationX, destinationY, status, numRiders, cardNumber, customerID)

VALUES ('TRP004', 8.9, '00:22:00', '40.7291N', '73.9965W', '40.7488N', '73.9680W', 'inprogress', 2, '4222222222222222', '0000000004');

INSERT INTO Trip (tripID, distance, duration, pickupX, pickupY, destinationX, destinationY, status, numRiders, cardNumber, customerID)

VALUES ('TRP005', 3.2, '00:10:00', '40.7488N', '73.9680W', '40.7128N', '74.0060W', 'completed', 4, '378282246310005', '0000000005');

INSERT INTO Trip (tripID, distance, duration, pickupX, pickupY, destinationX, destinationY, status, numRiders, cardNumber, customerID)

VALUES ('TRP006', 15.6, '00:40:00', '40.7128N', '74.0060W', '40.7306N', '73.9352W', 'inprogress', 1, '6011123456789010', '0000000006');

INSERT INTO Trip (tripID, distance, duration, pickupX, pickupY, destinationX, destinationY, status, numRiders, cardNumber, customerID)

VALUES ('TRP007', 9.8, '00:18:00', '40.7580N', '73.9855W', '40.7291N', '73.9965W', 'cancelled', 3, '5111111111111111', '0000000007');

INSERT INTO Trip (tripID, distance, duration, pickupX, pickupY, destinationX, destinationY, status, numRiders, cardNumber, customerID)

VALUES ('TRP008', 6.1, '00:17:00', '40.7291N', '73.9965W', '40.7488N', '73.9680W', 'completed', 2, '5105101051055100', '0000000008');

INSERT INTO Trip (tripID, distance, duration, pickupX, pickupY, destinationX, destinationY, status, numRiders, cardNumber, customerID)

VALUES ('TRP009', 11.4, '00:30:00', '40.7488N', '73.9680W', '40.7128N', '74.0060W', 'completed', 4, '4485485485485485', '0000000009');

INSERT INTO Trip (tripID, distance, duration, pickupX, pickupY, destinationX, destinationY, status, numRiders, cardNumber, customerID)

VALUES ('TRP010', 4.7, '00:12:00', '40.7306N', '73.9352W', '40.7580N', '73.9855W', 'inprogress', 1, '4111111111111111', '000000000A');

-- Inserting data into MaintenanceStaff Table

INSERT INTO MaintenanceStaff (staffID, firstName, lastName, dateOfEmployment, salary, phone, email)

VALUES ('1', 'John', 'Doe', TO\_DATE('2020-05-15', 'YYYY-MM-DD'), 55000, '123-456-7890', 'john.doe@example.com');

INSERT INTO MaintenanceStaff (staffID, firstName, lastName, dateOfEmployment, salary, phone, email) VALUES ('2', 'Jane', 'Smith', TO\_DATE('2019-08-22', 'YYYY-MM-DD'), 60000, '234-567-8901',

'jane.smith@example.com');

INSERT INTO MaintenanceStaff (staffID, firstName, lastName, dateOfEmployment, salary, phone, email) VALUES ('3', 'Alice', 'Brown', TO\_DATE('2021-07-30', 'YYYY-MM-DD'), 45000, '345-678-9012', 'alice.brown@example.com');

INSERT INTO MaintenanceStaff (staffID, firstName, lastName, dateOfEmployment, salary, phone, email) VALUES ('4', 'Mike', 'Johnson', TO\_DATE('2018-12-12', 'YYYY-MM-DD'), 70000, '456-789-0123', 'mike.johnson@example.com');

INSERT INTO MaintenanceStaff (staffID, firstName, lastName, dateOfEmployment, salary, phone, email) VALUES ('5', 'Chris', 'Davis', TO\_DATE('2022-03-17', 'YYYY-MM-DD'), 40000, '567-890-1234',

'chris.davis@example.com');

INSERT INTO MaintenanceStaff (staffID, firstName, lastName, dateOfEmployment, salary, phone, email) VALUES ('6', 'Sarah', 'Wilson', TO\_DATE('2017-11-25', 'YYYY-MM-DD'), 65000, '678-901-2345', 'sarah.wilson@example.com');

INSERT INTO MaintenanceStaff (staffID, firstName, lastName, dateOfEmployment, salary, phone, email) VALUES ('7', 'David', 'Martinez', TO\_DATE('2016-06-06', 'YYYY-MM-DD'), 55000, '789-012-3456', 'david.martinez@example.com');

INSERT INTO MaintenanceStaff (staffID, firstName, lastName, dateOfEmployment, salary, phone, email)

VALUES ('8', 'Anna', 'Taylor', TO\_DATE('2023-01-15', 'YYYY-MM-DD'), 50000, '890-123-4567', 'anna.taylor@example.com');

INSERT INTO MaintenanceStaff (staffID, firstName, lastName, dateOfEmployment, salary, phone, email)

VALUES ('9', 'Robert', 'Lee', TO\_DATE('2021-09-28', 'YYYY-MM-DD'), 62000, '901-234-5678', 'robert.lee@example.com');

INSERT INTO MaintenanceStaff (staffID, firstName, lastName, dateOfEmployment, salary, phone, email) VALUES ('10', 'Linda', 'Harris', TO\_DATE('2020-04-09', 'YYYY-MM-DD'), 52000, '012-345-6789', 'linda.harris@example.com');

INSERT INTO MaintenanceProducts (productID, productName, unitPrice, quantityInStock) VALUES ('101', 'Cleaning Solution', 15.99, 100);

INSERT INTO MaintenanceProducts (productID, productName, unitPrice, quantityInStock) VALUES ('102', 'Lubricant Oil', 8.50, 50);

INSERT INTO MaintenanceProducts (productID, productName, unitPrice, quantityInStock) VALUES ('103', 'Wrench Set', 25.00, 30);

INSERT INTO MaintenanceProducts (productID, productName, unitPrice, quantityInStock) VALUES ('104', 'Screwdriver Kit', 12.75, 200);

INSERT INTO MaintenanceProducts (productID, productName, unitPrice, quantityInStock) VALUES ('105', 'Hammer', 10.50, 60);

INSERT INTO MaintenanceProducts (productID, productName, unitPrice, quantityInStock) VALUES ('106', 'Safety Gloves', 5.99, 150);

INSERT INTO MaintenanceProducts (productID, productName, unitPrice, quantityInStock) VALUES ('107', 'Face Mask', 2.50, 300);

INSERT INTO MaintenanceProducts (productID, productName, unitPrice, quantityInStock) VALUES ('108', 'Toolbox', 45.99, 25);

INSERT INTO MaintenanceProducts (productID, productName, unitPrice, quantityInStock) VALUES ('109', 'Grease', 9.99, 80);

INSERT INTO MaintenanceProducts (productID, productName, unitPrice, quantityInStock) VALUES ('110', 'Pliers', 14.49, 75);

--Taxi Table Inserts

INSERT INTO Taxi (taxiID, year, make, model, capacity, spotNumber, currentX, currentY, inComission) VALUES ('TX001', 2016, 'Chrysler', 'Pacifica', 7, 45, 69000, 6000, 'Yes');

INSERT INTO Taxi (taxiID, year, make, model, capacity, spotNumber, currentX, currentY, inComission) VALUES ('TX002', 2005, 'Ford', 'Crown Victoria', 4, 12, 60500, 5100, 'Yes');

INSERT INTO Taxi (taxiID, year, make, model, capacity, spotNumber, currentX, currentY, inComission) VALUES ('TX003', 2010, 'Toyota', 'Prius', 4, 15, 45000, 3000, 'Yes');

INSERT INTO Taxi (taxiID, year, make, model, capacity, spotNumber, currentX, currentY, inComission) VALUES ('TX004', 2012, 'Chevrolet', 'Impala', 4, 9, 71000, 5200, 'No');

INSERT INTO Taxi (taxiID, year, make, model, capacity, spotNumber, currentX, currentY, inComission) VALUES ('TX005', 2018, 'Honda', 'Civic', 4, 3, 23000, 2700, 'Yes');

INSERT INTO Taxi (taxiID, year, make, model, capacity, spotNumber, currentX, currentY, inComission) VALUES

('TX006', 2020, 'Tesla', 'Model 3', 4, 18, 15000, 2500, 'Yes');

INSERT INTO Taxi (taxiID, year, make, model, capacity, spotNumber, currentX, currentY, inComission) VALUES ('TX007', 2016, 'Nissan', 'Altima', 4, 5, 54000, 4100, 'No');

INSERT INTO Taxi (taxiID, year, make, model, capacity, spotNumber, currentX, currentY, inComission) VALUES

('TX008', 2017, 'Hyundai', 'Sonata', 4, 6, 32000, 3900, 'Yes');

INSERT INTO Taxi (taxiID, year, make, model, capacity, spotNumber, currentX, currentY, inComission) VALUES ('TX009', 2015, 'Kia', 'Optima', 4, 8, 55000, 4800, 'No');

INSERT INTO Taxi (taxiID, year, make, model, capacity, spotNumber, currentX, currentY, inComission) VALUES

('TX010', 2019, 'Volkswagen', 'Jetta', 4, 10, 26000, 3200, 'Yes');

INSERT INTO MaintenanceWorkOrder (orderID, dateCreated, dueDate, dateFulfilled, taxiID, adminID)

VALUES ('201', TO\_DATE('01-DEC-23', 'DD-MON-YY'), TO\_DATE('15-DEC-23', 'DD-MON-YY'), TO\_DATE('14-DEC-23', 'DD-MON-YY'), 'TX001', 'ADM001');

INSERT INTO MaintenanceWorkOrder (orderID, dateCreated, dueDate, dateFulfilled, taxiID, adminID)

VALUES ('202', TO\_DATE('05-DEC-23', 'DD-MON-YY'), TO\_DATE('20-DEC-23', 'DD-MON-YY'), TO\_DATE('19-DEC-23', 'DD-MON-YY'), 'TX002', 'ADM002');

INSERT INTO MaintenanceWorkOrder (orderID, dateCreated, dueDate, dateFulfilled, taxiID, adminID)

VALUES ('203', TO\_DATE('10-DEC-23', 'DD-MON-YY'), TO\_DATE('25-DEC-23', 'DD-MON-YY'), TO\_DATE('24-DEC-23', 'DD-MON-YY'), 'TX003', 'ADM003');

INSERT INTO MaintenanceWorkOrder (orderID, dateCreated, dueDate, dateFulfilled, taxiID, adminID)

VALUES ('204', TO\_DATE('15-DEC-23', 'DD-MON-YY'), TO\_DATE('30-DEC-23', 'DD-MON-YY'), TO\_DATE('29-DEC-23', 'DD-MON-YY'), 'TX004', 'ADM001');

INSERT INTO MaintenanceWorkOrder (orderID, dateCreated, dueDate, dateFulfilled, taxiID, adminID)

VALUES ('205', TO\_DATE('20-DEC-23', 'DD-MON-YY'), TO\_DATE('05-JAN-24', 'DD-MON-YY'), TO\_DATE('04-JAN-24', 'DD-MON-YY'), 'TX005', 'ADM002');

INSERT INTO MaintenanceWorkOrder (orderID, dateCreated, dueDate, dateFulfilled, taxiID, adminID)

VALUES ('206', TO\_DATE('01-JAN-24', 'DD-MON-YY'), TO\_DATE('15-JAN-24', 'DD-MON-YY'), TO\_DATE('14-JAN-24', 'DD-MON-YY'), 'TX006', 'ADM003');

INSERT INTO MaintenanceWorkOrder (orderID, dateCreated, dueDate, dateFulfilled, taxiID, adminID)

VALUES ('207', TO\_DATE('05-JAN-24', 'DD-MON-YY'), TO\_DATE('20-JAN-24', 'DD-MON-YY'), TO\_DATE('19-JAN-24', 'DD-MON-YY'), 'TX007', 'ADM001');

INSERT INTO MaintenanceWorkOrder (orderID, dateCreated, dueDate, dateFulfilled, taxiID, adminID)

VALUES ('208', TO\_DATE('10-JAN-24', 'DD-MON-YY'), TO\_DATE('25-JAN-24', 'DD-MON-YY'), TO\_DATE('24-JAN-24', 'DD-MON-YY'), 'TX008', 'ADM002');

INSERT INTO MaintenanceWorkOrder (orderID, dateCreated, dueDate, dateFulfilled, taxiID, adminID)

VALUES ('209', TO\_DATE('15-JAN-24', 'DD-MON-YY'), TO\_DATE('30-JAN-24', 'DD-MON-YY'), TO\_DATE('29-JAN-24',

'DD-MON-YY'), 'TX009', 'ADM003');

INSERT INTO MaintenanceWorkOrder (orderID, dateCreated, dueDate, dateFulfilled, taxiID, adminID)

VALUES ('210', TO\_DATE('20-JAN-24', 'DD-MON-YY'), TO\_DATE('05-FEB-24', 'DD-MON-YY'), TO\_DATE('04-FEB-24', 'DD-MON-YY'), 'TX010', 'ADM001');

-- Inserting data into Use Table (Assuming valid foreign keys exist in MaintenanceProducts and MaintenanceWorkOrder) INSERT INTO Use (productID, orderID) VALUES ('101', '201');

INSERT INTO Use (productID, orderID) VALUES ('102', '202');

INSERT INTO Use (productID, orderID) VALUES ('103', '203');

INSERT INTO Use (productID, orderID) VALUES ('104', '204');

INSERT INTO Use (productID, orderID) VALUES ('105', '205');

INSERT INTO Use (productID, orderID) VALUES ('106', '206');

INSERT INTO Use (productID, orderID) VALUES ('107', '207');

INSERT INTO Use (productID, orderID) VALUES ('108', '208');

INSERT INTO Use (productID, orderID) VALUES ('109', '209');

INSERT INTO Use (productID, orderID) VALUES ('110', '210');

INSERT INTO AssignedTo (staffID, orderID) VALUES ('1', '201');

INSERT INTO AssignedTo (staffID, orderID) VALUES ('2', '202');

INSERT INTO AssignedTo (staffID, orderID) VALUES ('3', '203');

INSERT INTO AssignedTo (staffID, orderID) VALUES ('4', '204');

INSERT INTO AssignedTo (staffID, orderID) VALUES ('5', '205');

INSERT INTO AssignedTo (staffID, orderID) VALUES ('6', '206');

INSERT INTO AssignedTo (staffID, orderID) VALUES ('7', '207');

INSERT INTO AssignedTo (staffID, orderID) VALUES ('8', '208');

INSERT INTO AssignedTo (staffID, orderID) VALUES ('9', '209');

INSERT INTO AssignedTo (staffID, orderID) VALUES ('10', '210');

--CustomerPaymentInfoLink inserts

INSERT INTO CustomerPaymentInfoLink (customerID, cardNumber) VALUES ('0000000001', '4532123456789010');

INSERT INTO CustomerPaymentInfoLink (customerID, cardNumber) VALUES ('0000000002', '4012345678901234');

INSERT INTO CustomerPaymentInfoLink (customerID, cardNumber) VALUES ('0000000003', '6011234567890123');

INSERT INTO CustomerPaymentInfoLink (customerID, cardNumber) VALUES ('0000000004', '4222222222222222');

INSERT INTO CustomerPaymentInfoLink (customerID, cardNumber) VALUES ('0000000005', '378282246310005');

INSERT INTO CustomerPaymentInfoLink (customerID, cardNumber) VALUES ('0000000006', '6011123456789010');

INSERT INTO CustomerPaymentInfoLink (customerID, cardNumber) VALUES ('0000000007', '5111111111111111');

INSERT INTO CustomerPaymentInfoLink (customerID, cardNumber) VALUES ('0000000008', '5105101051055100');

INSERT INTO CustomerPaymentInfoLink (customerID, cardNumber) VALUES ('0000000009', '4111111111111111');

INSERT INTO CustomerPaymentInfoLink (customerID, cardNumber) VALUES ('000000000A', '4485485485485485'); ---Car Insurance Inserts

INSERT INTO CarInsurance (policyNumber, taxiID, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL123', 'TX001', TO\_DATE('01-JAN-24', 'DD-MON-YY'), TO\_DATE('01-JAN-25', 'DD-MON-YY'), 'InsuranceCo', 150.00);

INSERT INTO CarInsurance (policyNumber, taxiID, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL124', 'TX002', TO\_DATE('15-FEB-24', 'DD-MON-YY'), TO\_DATE('15-FEB-25', 'DD-MON-YY'), 'BestInsurer', 175.50);

INSERT INTO CarInsurance (policyNumber, taxiID, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL125', 'TX003', TO\_DATE('10-MAR-24', 'DD-MON-YY'), TO\_DATE('10-MAR-25', 'DD-MON-YY'), 'SecureCoverage', 160.00);

INSERT INTO CarInsurance (policyNumber, taxiID, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL126', 'TX004', TO\_DATE('05-APR-24', 'DD-MON-YY'), TO\_DATE('05-APR-25', 'DD-MON-YY'), 'ReliableInsure', 140.00);

INSERT INTO CarInsurance (policyNumber, taxiID, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL127', 'TX005', TO\_DATE('20-MAY-24', 'DD-MON-YY'), TO\_DATE('20-MAY-25', 'DD-MON-YY'), 'FastInsurance', 155.00);

INSERT INTO CarInsurance (policyNumber, taxiID, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL128', 'TX006', TO\_DATE('30-JUN-24', 'DD-MON-YY'), TO\_DATE('30-JUN-25', 'DD-MON-YY'), 'TrustyInsure', 165.00);

INSERT INTO CarInsurance (policyNumber, taxiID, startDate, expirationDate, companyName, monthlyRate) VALUES ('POL129', 'TX007', TO\_DATE('25-JUL-24', 'DD-MON-YY'), TO\_DATE('25-JUL-25', 'DD-MON-YY'), 'PremierCoverage', 180.00);

INSERT INTO CarInsurance (policyNumber, taxiID, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL130', 'TX008', TO\_DATE('15-AUG-24', 'DD-MON-YY'), TO\_DATE('15-AUG-25', 'DD-MON-YY'), 'GlobalInsurers', 170.00);

INSERT INTO CarInsurance (policyNumber, taxiID, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL131', 'TX009', TO\_DATE('01-SEP-24', 'DD-MON-YY'), TO\_DATE('01-SEP-25', 'DD-MON-YY'), 'SureSafe', 145.00);

INSERT INTO CarInsurance (policyNumber, taxiID, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL132', 'TX010', TO\_DATE('20-OCT-24', 'DD-MON-YY'), TO\_DATE('20-OCT-25', 'DD-MON-YY'), 'EliteCoverage', 160.00);

--Accidents Inserts

INSERT INTO Accidents (accidentID, incidentDate, locationX, locationY, description, taxiID)

VALUES ('ACC123', TO\_DATE('01-JAN-24', 'DD-MON-YY'), 123, 2000, 'Front side collision', 'TX002');

INSERT INTO Accidents (accidentID, incidentDate, locationX, locationY, description, taxiID)

VALUES ('ACC124', TO\_DATE('05-JAN-24', 'DD-MON-YY'), 150, 2050, 'Rear-end collision', 'TX003');

INSERT INTO Accidents (accidentID, incidentDate, locationX, locationY, description, taxiID)

VALUES ('ACC125', TO\_DATE('10-JAN-24', 'DD-MON-YY'), 200, 2100, 'Side swipe accident', 'TX004');

INSERT INTO Accidents (accidentID, incidentDate, locationX, locationY, description, taxiID)

VALUES ('ACC126', TO\_DATE('15-JAN-24', 'DD-MON-YY'), 250, 2150, 'Hit and run incident', 'TX005');

INSERT INTO Accidents (accidentID, incidentDate, locationX, locationY, description, taxiID)

VALUES ('ACC127', TO\_DATE('20-JAN-24', 'DD-MON-YY'), 300, 2200, 'Single vehicle accident', 'TX006');

INSERT INTO Accidents (accidentID, incidentDate, locationX, locationY, description, taxiID)

VALUES ('ACC128', TO\_DATE('25-JAN-24', 'DD-MON-YY'), 350, 2250, 'Intersection collision', 'TX007');

INSERT INTO Accidents (accidentID, incidentDate, locationX, locationY, description, taxiID)

VALUES ('ACC129', TO\_DATE('30-JAN-24', 'DD-MON-YY'), 400, 2300, 'Road obstruction incident', 'TX008');

INSERT INTO Accidents (accidentID, incidentDate, locationX, locationY, description, taxiID)

VALUES ('ACC130', TO\_DATE('05-FEB-24', 'DD-MON-YY'), 450, 2350, 'Weather-related accident', 'TX009');

INSERT INTO Accidents (accidentID, incidentDate, locationX, locationY, description, taxiID)

VALUES ('ACC131', TO\_DATE('10-FEB-24', 'DD-MON-YY'), 500, 2400, 'Collision with a pedestrian', 'TX010');

INSERT INTO Accidents (accidentID, incidentDate, locationX, locationY, description, taxiID)

VALUES ('ACC132', TO\_DATE('15-FEB-24', 'DD-MON-YY'), 550, 2450, 'Accident with a bicycle', 'TX002');

INSERT INTO Accidents (accidentID, incidentDate, locationX, locationY, description, taxiID)

VALUES ('ACC133', TO\_DATE('20-FEB-24', 'DD-MON-YY'), 600, 2500, 'Multi-vehicle collision', 'TX003');

-- Inserting data into DrivingAssignments Table

INSERT INTO DrivingAssignments (assignmentID, status, startDateTime, endDateTime, staffID, driverID, taxiID)

VALUES ('ASSG001', 'active', TO\_TIMESTAMP('2023-11-01 08:00', 'YYYY-MM-DD HH24:MI'), TO\_TIMESTAMP('202311-01 12:00', 'YYYY-MM-DD HH24:MI'), 'CAS001', 'DRV001', 'TX001');

INSERT INTO DrivingAssignments (assignmentID, status, startDateTime, endDateTime, staffID, driverID, taxiID)

VALUES ('ASSG002', 'completed', TO\_TIMESTAMP('2023-11-02 09:00', 'YYYY-MM-DD HH24:MI'), TO\_TIMESTAMP('2023-11-02 13:00', 'YYYY-MM-DD HH24:MI'), 'CAS002', 'DRV002', 'TX002');

INSERT INTO DrivingAssignments (assignmentID, status, startDateTime, endDateTime, staffID, driverID, taxiID)

VALUES ('ASSG003', 'pending', TO\_TIMESTAMP('2023-11-03 07:00', 'YYYY-MM-DD HH24:MI'), TO\_TIMESTAMP('2023-11-03 11:00', 'YYYY-MM-DD HH24:MI'), 'CAS003', 'DRV003', 'TX003');

INSERT INTO DrivingAssignments (assignmentID, status, startDateTime, endDateTime, staffID, driverID, taxiID)

VALUES ('ASSG004', 'active', TO\_TIMESTAMP('2023-11-04 10:00', 'YYYY-MM-DD HH24:MI'), TO\_TIMESTAMP('202311-04 14:00', 'YYYY-MM-DD HH24:MI'), 'CAS004', 'DRV004', 'TX004');

INSERT INTO DrivingAssignments (assignmentID, status, startDateTime, endDateTime, staffID, driverID, taxiID)

VALUES ('ASSG005', 'completed', TO\_TIMESTAMP('2023-11-05 11:00', 'YYYY-MM-DD HH24:MI'),

TO\_TIMESTAMP('2023-11-05 15:00', 'YYYY-MM-DD HH24:MI'), 'CAS005', 'DRV005', 'TX005');

INSERT INTO DrivingAssignments (assignmentID, status, startDateTime, endDateTime, staffID, driverID, taxiID)

VALUES ('ASSG006', 'pending', TO\_TIMESTAMP('2023-11-06 12:00', 'YYYY-MM-DD HH24:MI'), TO\_TIMESTAMP('2023-11-06 16:00', 'YYYY-MM-DD HH24:MI'), 'CAS006', 'DRV006', 'TX006');

INSERT INTO DrivingAssignments (assignmentID, status, startDateTime, endDateTime, staffID, driverID, taxiID)

VALUES ('ASSG007', 'active', TO\_TIMESTAMP('2023-11-07 08:00', 'YYYY-MM-DD HH24:MI'), TO\_TIMESTAMP('202311-07 12:00', 'YYYY-MM-DD HH24:MI'), 'CAS007', 'DRV007', 'TX007');

INSERT INTO DrivingAssignments (assignmentID, status, startDateTime, endDateTime, staffID, driverID, taxiID)

VALUES ('ASSG008', 'completed', TO\_TIMESTAMP('2023-11-08 09:00', 'YYYY-MM-DD HH24:MI'), TO\_TIMESTAMP('2023-11-08 13:00', 'YYYY-MM-DD HH24:MI'), 'CAS008', 'DRV008', 'TX008');

INSERT INTO DrivingAssignments (assignmentID, status, startDateTime, endDateTime, staffID, driverID, taxiID)

VALUES ('ASSG009', 'active', TO\_TIMESTAMP('2023-11-09 07:00', 'YYYY-MM-DD HH24:MI'), TO\_TIMESTAMP('202311-09 11:00', 'YYYY-MM-DD HH24:MI'), 'CAS009', 'DRV009', 'TX009');

INSERT INTO DrivingAssignments (assignmentID, status, startDateTime, endDateTime, staffID, driverID, taxiID)

VALUES ('ASSG010', 'pending', TO\_TIMESTAMP('2023-11-10 10:00', 'YYYY-MM-DD HH24:MI'), TO\_TIMESTAMP('2023-11-10 14:00', 'YYYY-MM-DD HH24:MI'), 'CAS010', 'DRV010', 'TX010');

## UPDATES.txt

-- Updating data in Administrator Table

UPDATE Administrator

SET salary = 65000

WHERE adminID = 'ADM001';

UPDATE Administrator

SET email = 'john.doe.updated@example.com'

WHERE adminID = 'ADM001';

UPDATE Administrator

SET phone = '0987654321'

WHERE adminID = 'ADM002';

UPDATE Administrator

SET salary = salary + 5000

WHERE adminID = 'ADM003';

-- Updating data in CustomerAssistanceStaff Table

UPDATE CustomerAssistanceStaff

SET salary = 48000

WHERE staffID = 'CAS001';

UPDATE CustomerAssistanceStaff

SET email = 'matthew.green.updated@example.com'

WHERE staffID = 'CAS001';

UPDATE CustomerAssistanceStaff

SET phone = '5559998888'

WHERE staffID = 'CAS002';

UPDATE CustomerAssistanceStaff

SET dateOfEmployment = TO\_TIMESTAMP('2023-01-15', 'YYYY-MM-DD')

WHERE staffID = 'CAS003';

-- Updating data in DrivingAssignments Table

UPDATE DrivingAssignments

SET status = 'completed'

WHERE assignmentID = 'ASSG001';

UPDATE DrivingAssignments

SET endDateTime = TO\_TIMESTAMP('2023-11-01 13:00', 'YYYY-MM-DD HH24:MI')

WHERE assignmentID = 'ASSG001';

UPDATE DrivingAssignments

SET startDateTime = TO\_TIMESTAMP('2023-11-02 10:00', 'YYYY-MM-DD HH24:MI')

WHERE assignmentID = 'ASSG002';

UPDATE DrivingAssignments

SET status = 'active'

WHERE assignmentID = 'ASSG003';

-- Update salary for a specific driver

UPDATE Driver

SET salary = 50000

WHERE driverID = 'DRV003';

-- Increase salary by 5% for drivers employed before 2021

UPDATE Driver

SET salary = salary \* 1.05

WHERE dateOfEmployment < TO\_DATE('2021-01-01', 'YYYY-MM-DD');

-- Update license number for drivers earning more than 55000

UPDATE Driver

SET licenseNumber = UPPER(licenseNumber)

WHERE salary > 55000;

-- Raise salary for driver with salary under 48k

UPDATE Driver

SET salary = salary \* 1.05

WHERE salary < 48000;

-- Update monthly rate for a specific policy

UPDATE DriverInsurance

SET monthlyRate = '160'

WHERE policyNumber = 'POL001';

-- Increase monthly rate by 10% for all policies with a rate below 150

UPDATE DriverInsurance

SET monthlyRate = monthlyRate \* 1.10

WHERE monthlyRate < 150;

-- Change the insurance company name for policies expiring in 2025

UPDATE DriverInsurance

SET companyName = 'FutureScr'

WHERE expirationDate BETWEEN TO\_DATE('2025-01-01', 'YYYY-MM-DD') AND TO\_DATE('2025-12-31', 'YYYY-MM-

DD');

-- Extend expiration date by one year for policies with monthly rate above 140

UPDATE DriverInsurance

SET expirationDate = expirationDate + INTERVAL '12 months'

WHERE monthlyRate > 140;

-------------------------------

-- Update trip status to 'completed' for trips with a duration of less than 15 minutes UPDATE Trip

SET status = 'completed'

WHERE duration < '00:15:00';

-- Increase distance by 10% for trips with more than 2 riders

UPDATE Trip

SET distance = distance \* 1.10

WHERE numRiders > 2;

-- Change the pickup location for a specific trip

UPDATE Trip

SET pickupX = '40.7350N', pickupY = '73.9900W'

WHERE tripID = 'TRP005';

-- Set status to 'cancelled' for all trips with a pickup location in New York coordinates (approximate) UPDATE Trip

SET status = 'cancelled'

WHERE pickupX LIKE '40.7%' AND pickupY LIKE '74.%';

-- Updating data in MaintenanceStaff Table

UPDATE MaintenanceStaff

SET salary = salary \* 1.05

WHERE staffID = '1';

UPDATE MaintenanceStaff

SET phone = '999-999-9999'

WHERE staffID = '2';

UPDATE MaintenanceStaff

SET email = 'updated.email@example.com'

WHERE staffID = '3';

UPDATE MaintenanceStaff

SET dateOfEmployment = TO\_DATE('2021-05-01', 'YYYY-MM-DD') WHERE staffID = '4';

-- Updating data in MaintenanceProducts Table

UPDATE MaintenanceProducts

SET unitPrice = unitPrice + 5

WHERE productID = '101';

UPDATE MaintenanceProducts

SET quantityInStock = quantityInStock - 20

WHERE productID = '102';

UPDATE MaintenanceProducts

SET productName = 'Advanced Wrench Set'

WHERE productID = '103';

UPDATE MaintenanceProducts

SET unitPrice = 20.00

WHERE productID = '104';

-- Updating data in Use Table (assuming the productID and orderID exist)

UPDATE Use

SET productID = 105

WHERE productID = '101' AND orderID = '201';

UPDATE Use

SET orderID = 210

WHERE productID = '102' AND orderID = '202';

UPDATE Use

SET productID = 106

WHERE productID = '103' AND orderID = '203';

UPDATE Use

SET orderID = 209

WHERE productID = '104' AND orderID = '204';

-- Updating data in AssignedTo Table (assuming the staffID and orderID exist)

UPDATE AssignedTo

SET staffID = 6

WHERE staffID = '1' AND orderID = '201';

UPDATE AssignedTo

SET orderID = 208

WHERE staffID = '2' AND orderID = '202';

UPDATE AssignedTo

SET staffID = 7

WHERE staffID = '3' AND orderID = '203';

UPDATE AssignedTo

SET orderID = 208

WHERE staffID = '4' AND orderID = '204';

--change the area code of all customer's named Sophia

UPDATE Customer

SET phone = CONCAT('813', SUBSTR(Customer.phone, 4))

WHERE Customer.firstName = 'Sophia';

--change format for emails in the example.com domain

UPDATE Customer

SET email = CONCAT(CONCAT(firstName, '.'), CONCAT(lastName, '@example.com'))

WHERE SUBSTR(email, STRPOS(email, '@') + 1) = 'example.com';

--change the last name of a specific customer

UPDATE Customer

SET lastName = 'Jenkins'

WHERE customerID = '0000000003';

--change the phone number of a specific customer

UPDATE Customer

SET phone = '727-638-0011'

WHERE customerID = '000000000A';

UPDATE PaymentInfo

SET CardName = 'Emily Jenkins'

WHERE CardNumber = '6011234567890123';

UPDATE PaymentInfo

SET addressLine1 = '1648 Leaf Lake Run', addressLine2 = NULL, PostalCode = '32801-1120'

WHERE CardNumber = '5105101051055100';

UPDATE PaymentInfo pi

SET expirationYear = expirationYear + 4

WHERE CardNumber = '4532123456789010';

UPDATE PaymentInfo pi

SET expirationYear = 2028

WHERE expirationYear < 2024;

UPDATE PostalCode pc

SET city = 'Winter Park', state = 'FL'

WHERE PostalCode = '32801-1234';

UPDATE PostalCode pc

SET city = 'Los Angeles', state = 'CA'

WHERE PostalCode = '90001-5678';

UPDATE PostalCode pc

SET city = 'New York', state = 'NY'

WHERE PostalCode = '10001-9101';

UPDATE PostalCode pc

SET city = 'Chicago', state = 'IL'

WHERE PostalCode = '60601-1122';

--Update Data in Taxi Table

--Update Car Model and Year

UPDATE Taxi

SET year = 2017, model = 'Pacifica 2.0'

WHERE taxiID = 'TX001';

--Update Commission Status

UPDATE Taxi

SET inComission = 'No'

WHERE taxiID = 'TX005';

--Update Car Capacity

UPDATE Taxi

SET capacity = 8

WHERE taxiID = 'TX002';

--Update Car Location

UPDATE Taxi

SET currentX = 75500, currentY = 6500

WHERE taxiID = 'TX007';

--Update Data in CarInsurance

--Update Monthly Rate

UPDATE CarInsurance

SET monthlyRate = 160.00

WHERE policyNumber = 'POL123';

--Update Company Insurance Name

UPDATE CarInsurance

SET companyName = 'NewInsuranceCo'

WHERE policyNumber = 'POL124';

--Update Insurance rate and expiration date

UPDATE CarInsurance

SET expirationDate = TO\_DATE('01-FEB-26', 'DD-MON-YY'), monthlyRate = 170.00

WHERE policyNumber = 'POL125';

--Update new monthly rate

UPDATE CarInsurance

SET monthlyRate = 150.00

WHERE policyNumber IN ('POL127', 'POL128');

--Update Accidents

UPDATE Accidents

SET locationX = 250, locationY = 2155

WHERE accidentID = 'ACC125';

UPDATE Accidents

SET description = 'Front-end collision at traffic light'

WHERE accidentID = 'ACC123';

UPDATE Accidents

SET incidentDate = TO\_DATE('15-JAN-24', 'DD-MON-YY'), locationX = 400, locationY = 2300 WHERE accidentID = 'ACC129';

UPDATE Accidents

SET incidentDate = TO\_DATE('01-MAR-24', 'DD-MON-YY')

WHERE taxiID = 'TX001';

--Update MaintenanceWorkOrder

UPDATE MaintenanceWorkOrder

SET dueDate = TO\_DATE('17-DEC-23', 'DD-MON-YY')

WHERE orderID = 'MWO001';

UPDATE MaintenanceWorkOrder

SET dateFulfilled = TO\_DATE('15-DEC-23', 'DD-MON-YY')

WHERE orderID = 'MWO002';

UPDATE MaintenanceWorkOrder

SET dateFulfilled = TO\_DATE('10-JAN-24', 'DD-MON-YY')

WHERE taxiID = 'TX005';

UPDATE MaintenanceWorkOrder

SET adminID = 'ADM006'

WHERE orderID = 'MWO007';

## ConstraintsCheck.txt

-- Not Null Constraint Violation: Attempting to insert NULL into a NOT NULL column (phone) in Administrator

INSERT INTO Administrator (adminID, firstName, lastName, phone, dateOfEmployment, email, salary)

VALUES ('ADM012', 'NullTest', 'Admin', NULL, TO\_DATE('2023-02-05', 'YYYY-MM-DD'), 'nulltest@example.com', 65000);

-- Check Constraint Violation: Attempting to insert an invalid salary value (negative salary) in Administrator

INSERT INTO Administrator (adminID, firstName, lastName, phone, dateOfEmployment, email, salary)

VALUES ('ADM013', 'NegativeSalary', 'Admin', '9876543210', TO\_DATE('2023-03-01', 'YYYY-MM-DD'), 'negativesalary@example.com', -5000);

-- Referential Integrity Violation: Attempting to insert a non-existent adminID in CustomerAssistanceStaff INSERT INTO CustomerAssistanceStaff (staffID, firstName, lastName, phone, dateOfEmployment, email, salary, adminID)

VALUES ('CAS011', 'ForeignKeyTest', 'Staff', '1231231234', TO\_DATE('2022-07-15', 'YYYY-MM-DD'), 'foreignkey@example.com', 47000, 'NOEXIST');

-- Check Constraint Violation: Attempting to insert an invalid phone number (not 10 digits) in CustomerAssistanceStaff INSERT INTO CustomerAssistanceStaff (staffID, firstName, lastName, phone, dateOfEmployment, email, salary, adminID)

VALUES ('CAS013', 'InvalidPhone', 'Staff', '123', TO\_DATE('2022-12-01', 'YYYY-MM-DD'), 'invalidphone@example.com', 46000, 'ADM003');

-- Check Constraint Violation: Attempting to set endDateTime before startDateTime in DrivingAssignments

INSERT INTO DrivingAssignments (assignmentID, status, startDateTime, endDateTime, staffID, driverID, taxiID)

VALUES ('ASSG012', 'completed', TO\_TIMESTAMP('2023-11-16 14:00', 'YYYY-MM-DD HH24:MI'), TO\_TIMESTAMP('2023-11-16 10:00', 'YYYY-MM-DD HH24:MI'), 'CAS001', 'DRV002', 'TX002');

--------------------------------------------------------------------------------------------------------------

-- 1. Violate PRIMARY KEY constraint by attempting to insert a duplicate policyNumber in DriverInsurance

INSERT INTO DriverInsurance (policyNumber, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL001', TO\_DATE('2024-01-01', 'YYYY-MM-DD'), TO\_DATE('2025-01-01', 'YYYY-MM-DD'), 'TestCo',

'140.00');

-- 2. Violate CHECK constraint on salary by inserting a negative value in Driver

INSERT INTO Driver (driverID, salary, dateOfEmployment, licenseNumber, policyNumber) VALUES ('DRV011', -5000, TO\_DATE('2023-01-15', 'YYYY-MM-DD'), 'LIC011', 'POL002');

-- 3. Violate CHECK constraint on duration by setting duration outside 24-hour range in Trip

INSERT INTO Trip (tripID, distance, duration, pickupX, pickupY, destinationX, destinationY, status, numRiders, customerID)

VALUES ('TRP011', 15.0, '24:00:00', '40.7128N', '74.0060W', '40.7306N', '73.9352W', 'inprogress', 2, 'CUST001');

-- 4. Violate NOT NULL constraint by inserting a NULL policyNumber in Driver

INSERT INTO Driver (driverID, salary, dateOfEmployment, licenseNumber, policyNumber) VALUES ('DRV012', 52000, TO\_DATE('2023-05-10', 'YYYY-MM-DD'), 'LIC012', NULL);

-- 5. Violate FOREIGN KEY constraint by setting an invalid customerID in Trip

INSERT INTO Trip (tripID, distance, duration, pickupX, pickupY, destinationX, destinationY, status, numRiders, customerID)

VALUES ('TRP012', 12.0, '00:35:00', '40.7580N', '73.9855W', '40.7306N', '73.9352W', 'completed', 3, 'CUST999');

-- 6. Violate PRIMARY KEY constraint by inserting a duplicate tripID in Trip

INSERT INTO Trip (tripID, distance, duration, pickupX, pickupY, destinationX, destinationY, status, numRiders, customerID)

VALUES ('TRP001', 8.5, '00:22:00', '40.7291N', '73.9965W', '40.7580N', '73.9855W', 'inprogress', 2, 'CUST003');

-- 7. Violate CHECK constraint on monthlyRate by inserting a non-numeric value in DriverInsurance

INSERT INTO DriverInsurance (policyNumber, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL011', TO\_DATE('2024-06-01', 'YYYY-MM-DD'), TO\_DATE('2025-06-01', 'YYYY-MM-DD'), 'SafeLife', 'ABC');

-- 8. Violate CHECK constraint on expirationDate by setting it before startDate in DriverInsurance

INSERT INTO DriverInsurance (policyNumber, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL012', TO\_DATE('2024-06-01', 'YYYY-MM-DD'), TO\_DATE('2024-05-01', 'YYYY-MM-DD'), 'CoverNow', 145.00);

-- 9. Violate FOREIGN KEY constraint by setting an invalid policyNumber in Driver

INSERT INTO Driver (driverID, salary, dateOfEmployment, licenseNumber, policyNumber)

VALUES ('DRV014', 48000, TO\_DATE('2024-11-01', 'YYYY-MM-DD'), 'LIC014', 'POL999');



**Assignment 4: Data**

**&**

**Queries**

This assignment involved writing complex SQL queries to extract meaningful insights from the database. Queries included aggregations, joins, subqueries, and conditional filtering to address

various operational requirements like trip analytics, customer reports, and resource management.

## Queries.txt

-- Query 1: List all Customer Assistance Staff with their associated Administrators

-- This query retrieves Customer Assistance Staff members and the Administrators they report to.

SELECT CAS.staffID, CAS.firstName AS StaffFirstName, CAS.lastName AS StaffLastName,

A.firstName AS AdminFirstName, A.lastName AS AdminLastName

FROM CustomerAssistanceStaff CAS

JOIN Administrator A ON CAS.adminID = A.adminID;

-- Query 2: Find the total number of trips completed by each customer

-- Aggregates the number of trips completed by each customer, grouping by customerID.

SELECT C.customerID, C.firstName, C.lastName, COUNT(T.tripID) AS TotalTrips

FROM Customer C

JOIN Trip T ON C.customerID = T.customerID

WHERE T.status = 'completed'

GROUP BY C.customerID, C.firstName, C.lastName

HAVING COUNT(T.tripID) > 0;

-- Query 3: List all Maintenance Work Orders along with assigned staff and used products

-- Joins MaintenanceWorkOrder, AssignedTo, and Use tables to show details for each work order.

SELECT MWO.orderID, MWO.dateCreated, MS.firstName AS StaffFirstName, MP.productName

FROM MaintenanceWorkOrder MWO

JOIN AssignedTo AT ON MWO.orderID = AT.orderID

JOIN MaintenanceStaff MS ON AT.staffID = MS.staffID

JOIN Use U ON MWO.orderID = U.orderID

JOIN MaintenanceProducts MP ON U.productID = MP.productID;

-- Query 4: Find drivers with insurance policies that expire within the next 6 months -- Uses a subquery to filter policies expiring within the next 6 months.

SELECT D.driverID, D.licenseNumber, DI.companyName, DI.expirationDate

FROM Driver D

JOIN DriverInsurance DI ON D.policyNumber = DI.policyNumber

WHERE DI.expirationDate < (CURRENT\_DATE + INTERVAL '6 months');

-- Query 5: Calculate average salary of employees grouped by their role (Administrator, Customer Assistance, Maintenance)

-- Uses multiple joins with GROUP BY to calculate the average salary per role.

SELECT 'Administrator' AS Role, AVG(A.salary) AS AverageSalary

FROM Administrator A

UNION

SELECT 'Customer Assistance' AS Role, AVG(CAS.salary) AS AverageSalary

FROM CustomerAssistanceStaff CAS

UNION

SELECT 'Maintenance Staff' AS Role, AVG(MS.salary) AS AverageSalary FROM MaintenanceStaff MS;

-- Query 6

-- Retrieve each customer's name, their payment card's expiration year, and the total distance of completed trips by the customer using aggregates(SUM) . This query also joins 4 tables to display the information needed. This also checked the trip is completed.

SELECT c.firstName AS customer\_first\_name, c.lastName AS customer\_last\_name, pi.ExpirationYear AS payment\_card\_expiration\_year,

SUM(t.distance) AS total\_completed\_trip\_distance

FROM Customer c

JOIN CustomerPaymentInfoLink cpil ON c.customerID = cpil.customerID

JOIN PaymentInfo pi ON cpil.cardNumber = pi.CardNumber

JOIN Trip t ON c.customerID = t.customerID

WHERE t.status = 'completed'

GROUP BY c.customerID, c.firstName, c.lastName, pi.ExpirationYear;

--Query 7

-- Retrieve each administrator’s name, the total number of customer assistance staff they manage, and the total salary paid to these staff members. This query join Administrator and CustomerAssistanceStaff tables while using aggregates(SUM) to calculate total salary and aggregates(COUNT) to count the staff that report to specific admin.

SELECT a.firstName AS admin\_first\_name, a.lastName AS admin\_last\_name, COUNT(cas.staffID) AS staff\_count,

SUM(cas.salary) AS total\_staff\_salary

FROM Administrator a

JOIN CustomerAssistanceStaff cas ON a.adminID = cas.adminID

GROUP BY a.adminID, a.firstName, a.lastName;

-- find the average distance of trips taken in each city as -- indicated by the billing address of the card used for payment select pc.city, AVG(distance) as "Average Distance" from Customer c join Trip t on c.customerID = t.customerID join PaymentInfo pi on t.cardNumber = pi.CardNumber join PostalCode pc on pi.postalCode = pc.postalCode group by pc.city;

-- Select all maintenance staff who have worked on taxi TX001 select firstName, lastName, email from MaintenanceStaff m where exists( select taxiID

from AssignedTo a join MaintenanceWorkOrder mwo using(orderID) join Taxi t using(TaxiID) where taxiID = 'TX001' and m.staffID = a.staffID

);

--Show the Policy number, company name, and monthly rate of all insurances from DriverInsurance that are above average

--Uses WHERE and subquery

SELECT driverID, i.policyNumber, i.companyName, i.monthlyRate FROM DriverInsurance i

JOIN Driver d ON i.policyNumber = d.policyNumber

WHERE monthlyRate >= (SELECT AVG(monthlyRate) FROM DriverInsurance);

--Find the last taxi work order of 2023, show the order id, taxi id, model, policy number, date order was created and fulfilled

--uses where and subquery

SELECT taxiID, model, policyNumber, orderID, dateCreated, dateFulfilled

FROM Taxi

JOIN CarInsurance USING (taxiID)

JOIN MaintenanceWorkOrder USING (taxiID)

WHERE orderID = (SELECT MAX(orderID) FROM MaintenanceWorkOrder WHERE dateFulfilled <= TO\_DATE('31-DEC-23', 'DD-MON-YY'));



**Assignment 5: Views, Triggers & Stored Procedures**

The implementation of advanced database functionalities, including stored procedures, triggers, and views, enhanced automation and ensured the consistency of business rules. These features improved database performance and usability.

## SP.txt

-- Stored Procedure 1: procedure that performs anupdate on the driver table

-- This updates the salary of all drivers based on years of employment and retrieve updated details.

CREATE OR REPLACE FUNCTION UpdateDriverSalaries()

RETURNS VOID AS $$

BEGIN

UPDATE Driver

SET salary = salary \* 1.05

WHERE dateOfEmployment < NOW() - INTERVAL '3 years';

RAISE NOTICE 'Driver salaries updated for those employed more than 3 years.'; END;

$$ LANGUAGE plpgsql;

-- Stored Procedure 2: procedure that uses a conditional CASE statement to decide updates.

-- This updates the trip status based on the duration and rider count.

CREATE OR REPLACE FUNCTION UpdateTripStatus()

RETURNS VOID AS $$

BEGIN

UPDATE Trip

SET status = CASE

WHEN duration::interval < INTERVAL '00:15:00' THEN 'completed'

WHEN numRiders > 2 THEN 'inprogress'

ELSE 'pending'

END;

RAISE NOTICE 'Trip statuses updated.';

END;

$$ LANGUAGE plpgsql;

-- Stored Procdure 3: procedure that performs passing and receiving data.

-- This passes a customerID and returns the trip count.

CREATE OR REPLACE FUNCTION GetCustomerTrips(p\_customerID VARCHAR)

RETURNS INTEGER AS $$

DECLARE

p\_tripCount INTEGER;

BEGIN

SELECT COUNT(\*)

INTO p\_tripCount

FROM Trip

WHERE customerID = p\_customerID;

RAISE NOTICE 'Total trips for Customer %: %', p\_customerID, p\_tripCount; RETURN p\_tripCount;

END;

$$ LANGUAGE plpgsql;

-- Stored Procedure 4: Function that calculates the processed data and returns the total. So use of passing and recieving data and a function that processes it.

-- Processes data by calculating the total income for a customer using the SUM aggregate function. Input (p\_customerID) and returns a processed result (total income).

CREATE OR REPLACE FUNCTION GetCustomerIncome(p\_customerID VARCHAR)

RETURNS NUMERIC AS $$

DECLARE

totalIncome NUMERIC;

BEGIN

SELECT SUM(distance \* 2) -- Example rate per km

INTO totalIncome

FROM Trip

WHERE customerID = p\_customerID;

RETURN COALESCE(totalIncome, 0);

END;

$$ LANGUAGE plpgsql;

## TG.txt

--INSERT TRIGGER

--Will not allow the insert of a Driver who's salary is lower than 40,000

CREATE OR REPLACE TRIGGER MinInsertSalary

BEFORE INSERT ON Driver

FOR EACH ROW

DECLARE

minSalary INT := 40000; BEGIN

IF :NEW.salary < minSalary THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Salary must be at least $40,000.'); END IF;

END;

/

--DELETE Trigger

--Prevents deleting any Administrators that are currently in Customer Assistance Staff

CREATE OR REPLACE TRIGGER AdminDeleteinCAStaff

BEFORE DELETE ON Administrator

FOR EACH ROW

DECLARE

CAstaff\_count INT;

BEGIN

-- Count the number of staff linked to the administrator being deleted

SELECT COUNT(\*) INTO CAstaff\_count

FROM CustomerAssistanceStaff

WHERE adminID = :OLD.adminID;

-- If in CA staff, raise error

IF CAstaff\_count > 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Cannot delete an Administrator assigned to Customer Assistance Staff.'); END IF;

END;

/

-- UPDATE Trigger

-- Prevents updating the salary of a driver to below $40,000 by doing a comparision with minSalary.

CREATE OR REPLACE TRIGGER trg\_update\_driver\_salary

BEFORE UPDATE ON Driver

FOR EACH ROW

DECLARE

minSalary INT := 40000;

BEGIN

IF :NEW.salary < minSalary THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Salary cannot be reduced below $40,000.'); END IF;

END;

/

--FOR EACH ROWS TRIGGER

-- Prevent assigning a driver to a new driving assignment if they already have an active one

CREATE OR REPLACE TRIGGER trg\_prevent\_multiple\_driver\_assignments

BEFORE INSERT OR UPDATE ON DrivingAssignments

FOR EACH ROW DECLARE

v\_active\_assignments NUMBER;

BEGIN

-- Count the number of active assignments for the driver being assigned

SELECT COUNT(\*)

INTO v\_active\_assignments

FROM DrivingAssignments

WHERE driverID = :NEW.driverID

AND status = 'active'

AND (

:NEW.startDateTime BETWEEN startDateTime AND endDateTime

OR :NEW.endDateTime BETWEEN startDateTime AND endDateTime

);

-- If there is any active assignment overlapping with the new one, raise an error

IF v\_active\_assignments > 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Driver is already assigned to an active assignment during this time.'); END IF;

END;

/

## Views.txt

CREATE VIEW TaxiStats AS

SELECT ta.taxiID as taxiID, year, make, model, COUNT(da.assignmentID) as numAsignments, count(da.driverID) as numDrivers

FROM Taxi ta left JOIN DrivingAssignments da on ta.taxiID = da.taxiID GROUP BY ta.taxiID, year, make, model

CREATE VIEW CustomerStats AS

SELECT c.customerID as customerID, c.firstName as firstName, c.lastName as lastName, c.phone as phone, c.email as email, COUNT(t.tripID) as "TripsTaken",

Count(pi.Car dNumber) as "Payment Methods"

FROM Customer c LEFT JOIN Trip t on c.customerID = t.customerID LEFT JOIN CustomerPaymentInfoLink cpil on cpil.customerID = c.customerID join PaymentInfo pi on pi.CardNumber = cpil.cardNumber GROUP BY c.customerID, c.firstName, c.lastName, c.phone, c.email;

CREATE VIEW DriverStats AS

SELECT d.driverID, d.salary, count(da.assignmentID) as "numAssignments", count(t.taxiID) as "numTaxis" from Driver d left Join DrivingAssignments da on d.driverID = da.driverID join Taxi t on t.taxiID = da.taxiID group by d.driverID, d.salary

CREATE VIEW MaintenanceStaffView AS

SELECT staffID, firstName, lastName, dateOfEmployment from MaintenanceStaff

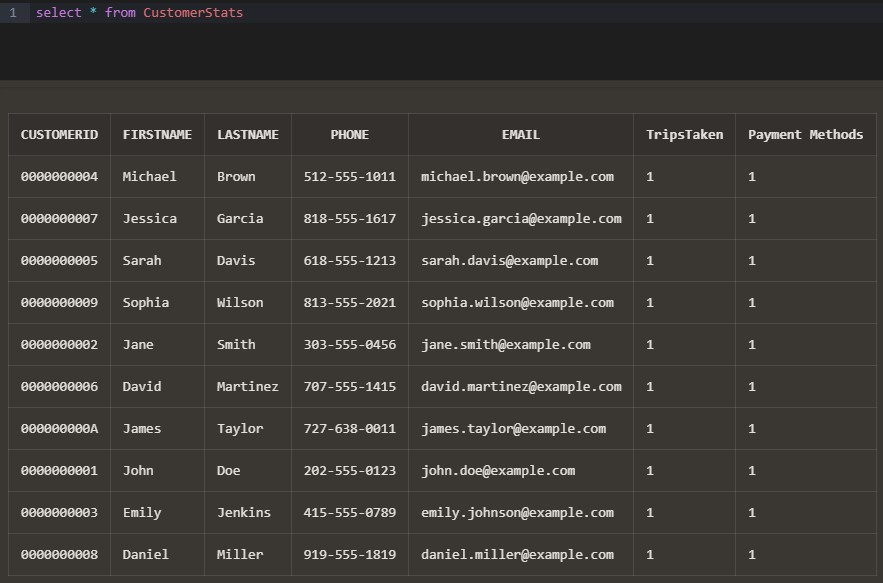
**EXAMPLES:**

VIEWS:

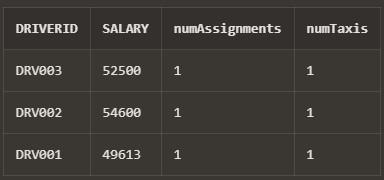
View – TaxiStats – Example (select \* from TaxiStats)



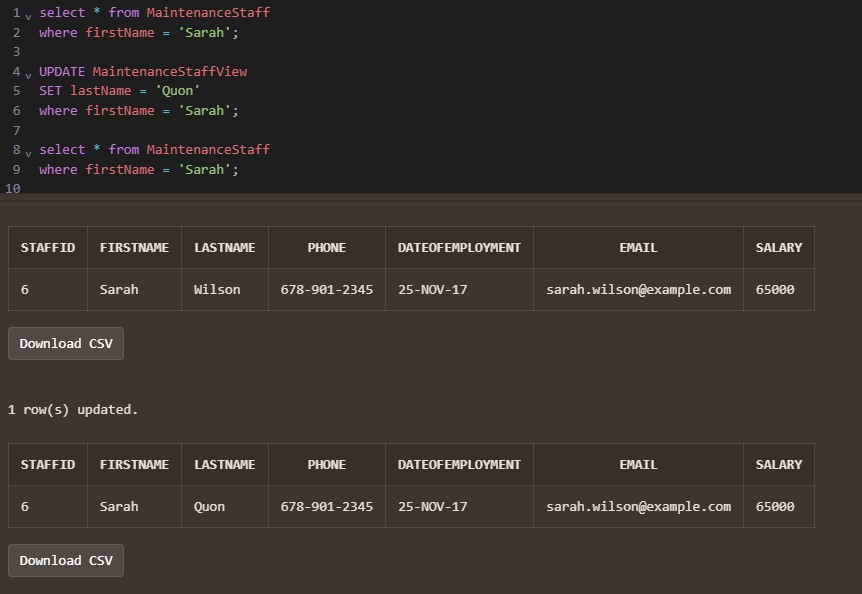
View – CustomerStats - Example



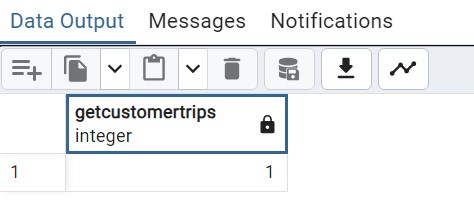
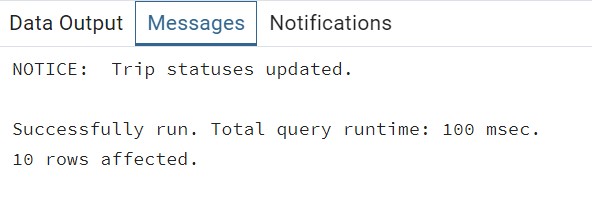
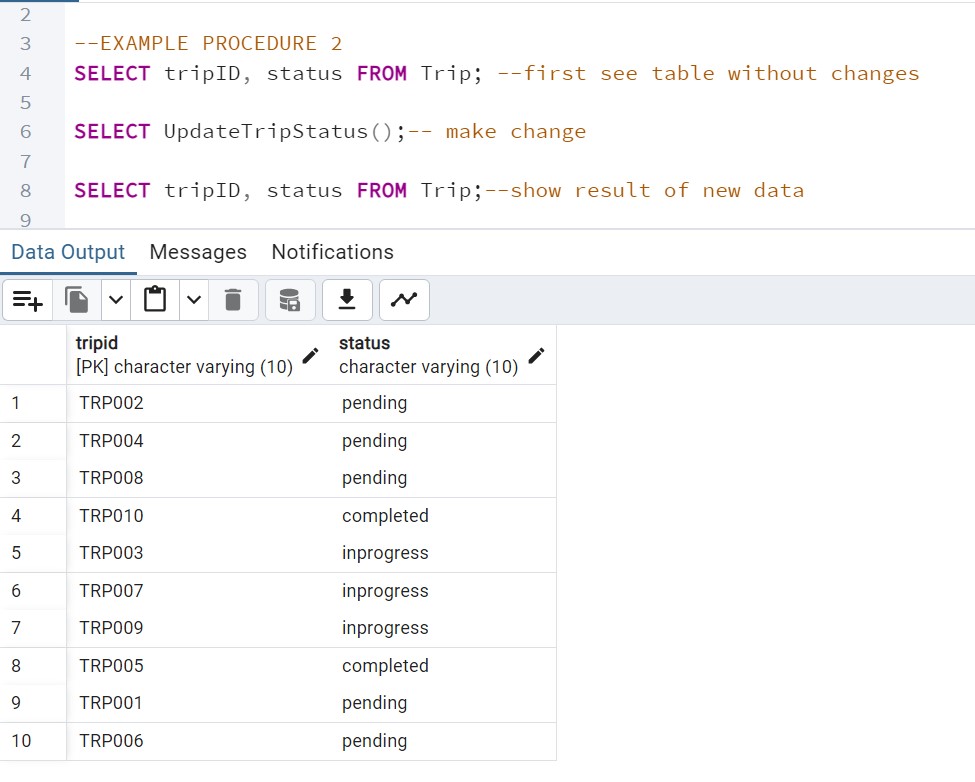
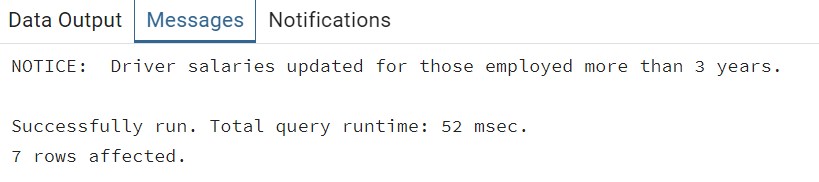
View – DriverStats – Example (select \* from DriverStats)



View – MaintenanceStaffView – Example of an update



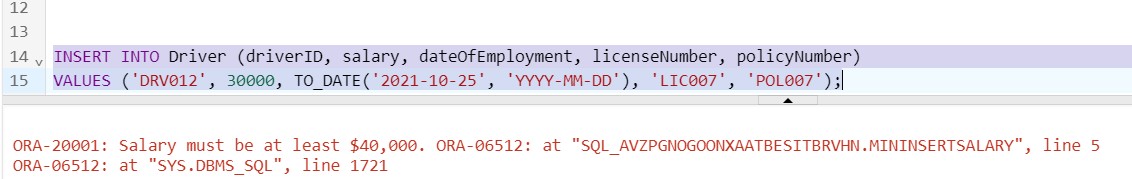
STORED PROCEDURES:



TRIGGERS

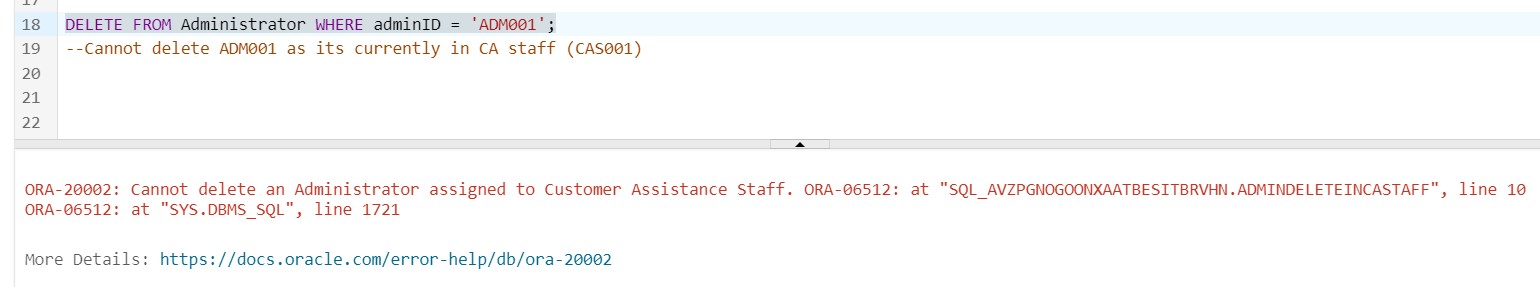
INSERT TRIGGER

--Insert will not work because MinInsertSalary Trigger won't accept salaries lower than 40000



DELETE TRIGGER

--Cannot delete ADM001 as its currently in CA staff (CAS001)



UPDATE TRIGGER

--Cannot Update Driver Salary lower than 40000



FOR EACH ROW TRIGGER

--Cannot insert a driving assignment when driver has an active assignment



**Assignment 6: Final Report**

The final assignment consolidated all previous work into a comprehensive report, including the information below:

**Rows Inserted into Each Table:**

* Administrator: 10 rows
* CustomerAssistanceStaff: 10 rows
* DriverInsurance: 10 rows
* Driver: 10 rows
* Customer: 10 rows
* PostalCode: 14 rows
* PaymentInfo: 10 rows
* Trip: 10 rows
* MaintenanceStaff: 10 rows
* MaintenanceProducts: 10 rows
* Taxi: 10 rows
* MaintenanceWorkOrder: 10 rows
* Use: 10 rows
* AssignedTo: 10 rows
* CustomerPaymentInfoLink: 10 rows
* CarInsurance: 10 rows
* Accidents: 11 rows
* Driving Assignments: 10 rows

**Most Interesting Updates and their results:**

1. Updating Trip Status for Short Duration Trips

-- Update trip status to 'completed' for trips with a duration of less than 15 minutes

SQL Statement:

UPDATE Trip

SET status = 'completed'

WHERE duration < '00:15:00';

Result: This update marks all trips with a duration of less than 15 minutes as "completed." This is significant because it automates status updates for quick trips, reducing manual intervention and ensuring accurate trip records.

1. Increasing Salary for Drivers Employed Before 2021 - - Increase salary by 5% for drivers employed before 2021

SQL Statement:

UPDATE Driver

SET salary = salary \* 1.05

WHERE dateOfEmployment < TO\_DATE('2021-01-01', 'YYYY-MM-DD');

Result: All drivers employed before 2021 received a 5% salary increase, recognizing their seniority. This not only boosts morale among experienced drivers but also ensures salary adjustments align with organizational policies.

1. Extending Insurance Expiration Date for High-Rate Policies

-- Extend expiration date by one year for policies with monthly rate above 140 SQL Statement:

UPDATE DriverInsurance

SET expirationDate = expirationDate + INTERVAL '12 months' WHERE monthlyRate > 140;

Result: Insurance policies with monthly rates above $140 were extended by one year. This ensures continued coverage for high-premium policies and avoids gaps in critical insurance services.

1. Updating Car Model and Year

--Update Car Model and Year

SQL Statement:

UPDATE Taxi

SET year = 2017, model = 'Pacifica 2.0'

WHERE taxiID = 'TX001';

Result: The model and year of taxi TX001 were updated to reflect an upgraded version ("Pacifica 2.0"). This update highlights efforts to modernize the fleet and maintain high service quality for customers.

**Constraint Check Results:**

1. NOT NULL Constraint Violation in Administrator

-- Attempting to insert NULL into a NOT NULL column (phone) in Administrator

SQL Statement:

INSERT INTO Administrator (adminID, firstName, lastName, phone, dateOfEmployment, email, salary)

VALUES ('ADM012', 'NullTest', 'Admin', NULL, TO\_DATE('2023-02-05', 'YYYY-MM-DD'), 'nulltest@example.com', 65000);

Result: The insertion failed because the phone column is defined as NOT NULL, and a NULL value was provided. This ensures essential data like contact information is always captured.

1. CHECK Constraint Violation for Negative Salary in Administrator

-- Check Constraint Violation: Attempting to insert an invalid salary value (negative salary) in Administrator

SQL Statement:

INSERT INTO Administrator (adminID, firstName, lastName, phone, dateOfEmployment, email, salary)

VALUES ('ADM013', 'NegativeSalary', 'Admin', '9876543210', TO\_DATE('2023-03-01', 'YYYY-MM-DD'),

'negativesalary@example.com', -5000);

Result: The insertion was rejected because the salary value violated the CHECK constraint that enforces non-negative salaries, maintaining logical data integrity.

1. FOREIGN KEY Constraint Violation in CustomerAssistanceStaff

-- Referential Integrity Violation: Attempting to insert a non-existent adminID in CustomerAssistanceStaff

SQL Statement:

INSERT INTO CustomerAssistanceStaff (staffID, firstName, lastName, phone, dateOfEmployment, email, salary, adminID)

VALUES ('CAS011', 'ForeignKeyTest', 'Staff', '1231231234', TO\_DATE('2022-07-15', 'YYYY-MM-DD'), 'foreignkey@example.com', 47000, 'NOEXIST');

Result: The insertion failed because adminID referenced a non-existent value in the Administrator table, ensuring referential integrity.

1. CHECK Constraint Violation for Invalid Phone Number in CustomerAssistanceStaff

-- Check Constraint Violation: Attempting to insert an invalid phone number (not 10 digits) in CustomerAssistanceStaff SQL Statement:

INSERT INTO CustomerAssistanceStaff (staffID, firstName, lastName, phone, dateOfEmployment, email, salary, adminID)

VALUES ('CAS013', 'InvalidPhone', 'Staff', '123', TO\_DATE('2022-12-01', 'YYYY-MM-DD'), 'invalidphone@example.com',

46000, 'ADM003');

Result: The insertion was blocked because the phone number did not meet the required length constraint, ensuring consistent data formatting.

1. CHECK Constraint Violation for Invalid endDateTime in DrivingAssignments

-- Check Constraint Violation: Attempting to set endDateTime before startDateTime in DrivingAssignments

SQL Statement:

INSERT INTO DrivingAssignments (assignmentID, status, startDateTime, endDateTime, staffID, driverID, taxiID)

VALUES ('ASSG012', 'completed', TO\_TIMESTAMP('2023-11-16 14:00', 'YYYY-MM-DD HH24:MI'), TO\_TIMESTAMP('2023-11-16 10:00', 'YYYY-MM-DD HH24:MI'), 'CAS001', 'DRV002', 'TX002');

Result: The insertion was rejected as endDateTime was earlier than startDateTime, violating the CHECK constraint that ensures logical time sequences.

1. PRIMARY KEY Violation in DriverInsurance

-- Violate PRIMARY KEY constraint by attempting to insert a duplicate policyNumber in DriverInsurance

SQL Statement:

INSERT INTO DriverInsurance (policyNumber, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL001', TO\_DATE('2024-01-01', 'YYYY-MM-DD'), TO\_DATE('2025-01-01', 'YYYY-MM-DD'), 'TestCo', '140.00');

Result: The insertion failed because POL001 already exists in the table, ensuring no duplicate entries for primary keys.

1. CHECK Constraint Violation for Negative Salary in Driver

-- Violate CHECK constraint on salary by inserting a negative value in Driver

SQL Statement:

INSERT INTO Driver (driverID, salary, dateOfEmployment, licenseNumber, policyNumber) VALUES ('DRV011', -5000, TO\_DATE('2023-01-15', 'YYYY-MM-DD'), 'LIC011', 'POL002');

Result: The insertion was blocked due to a negative salary, maintaining logical and valid financial data.

1. CHECK Constraint Violation for Invalid Duration in Trip

-- Violate CHECK constraint on duration by setting duration outside 24-hour range in Trip

SQL Statement:

INSERT INTO Trip (tripID, distance, duration, pickupX, pickupY, destinationX, destinationY, status, numRiders, customerID)

VALUES ('TRP011', 15.0, '24:00:00', '40.7128N', '74.0060W', '40.7306N', '73.9352W', 'inprogress', 2, 'CUST001');

Result: The insertion failed because the duration exceeded the valid range of a 24-hour period, enforcing valid data for trip durations.

1. FOREIGN KEY Violation for Invalid Policy in Driver

-- Violate NOT NULL constraint by inserting a NULL policyNumber in Driver

SQL Statement:

INSERT INTO Driver (driverID, salary, dateOfEmployment, licenseNumber, policyNumber)

VALUES ('DRV012', 52000, TO\_DATE('2023-05-10', 'YYYY-MM-DD'), 'LIC012', NULL);

Result: The insertion failed as the policyNumber referenced a non-existent record in the DriverInsurance table, maintaining referential integrity.

1. FOREIGN KEY Constraint Violation for Invalid Customer in Trip

-- Violate FOREIGN KEY constraint by setting an invalid customerID in Trip

SQL Statement:

INSERT INTO Trip (tripID, distance, duration, pickupX, pickupY, destinationX, destinationY, status, numRiders, customerID)

VALUES ('TRP012', 12.0, '00:35:00', '40.7580N', '73.9855W', '40.7306N', '73.9352W', 'completed', 3, 'CUST999');

Result: The insertion failed because CUST999 does not exist in the Customer table, enforcing referential integrity and ensuring only valid customer records are referenced.

1. PRIMARY KEY Violation in Trip

-- Violate PRIMARY KEY constraint by inserting a duplicate tripID in Trip

SQL Statement:

INSERT INTO Trip (tripID, distance, duration, pickupX, pickupY, destinationX, destinationY, status, numRiders, customerID)

VALUES ('TRP001', 8.5, '00:22:00', '40.7291N', '73.9965W', '40.7580N', '73.9855W', 'inprogress', 2, 'CUST003');

Result: The insertion was rejected because TRP001 already exists in the Trip table, ensuring unique trip identifiers.

1. CHECK Constraint Violation for Non-Numeric Monthly Rate in DriverInsurance

-- Violate CHECK constraint on monthlyRate by inserting a non-numeric value in DriverInsurance

SQL Statement:

INSERT INTO DriverInsurance (policyNumber, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL011', TO\_DATE('2024-06-01', 'YYYY-MM-DD'), TO\_DATE('2025-06-01', 'YYYY-MM-DD'), 'SafeLife', 'ABC');

Result: The insertion failed because the monthlyRate value was non-numeric, violating the CHECK constraint that enforces valid numerical values for rates.

1. CHECK Constraint Violation for Expiration Date in DriverInsurance

-- Violate CHECK constraint on expirationDate by setting it before startDate in DriverInsurance

SQL Statement:

INSERT INTO DriverInsurance (policyNumber, startDate, expirationDate, companyName, monthlyRate)

VALUES ('POL012', TO\_DATE('2024-06-01', 'YYYY-MM-DD'), TO\_DATE('2024-05-01', 'YYYY-MM-DD'), 'CoverNow', 145.00);

Result: The insertion failed as the expirationDate was earlier than the startDate, maintaining logical date relationships.

1. FOREIGN KEY Constraint Violation for Invalid Policy in Driver

-- Violate FOREIGN KEY constraint by setting an invalid policyNumber in Driver

SQL Statement:

INSERT INTO Driver (driverID, salary, dateOfEmployment, licenseNumber, policyNumber) VALUES ('DRV014', 48000, TO\_DATE('2024-11-01', 'YYYY-MM-DD'), 'LIC014', 'POL999');

Result: The insertion failed because POL999 does not exist in the DriverInsurance table, enforcing referential integrity.

**SQL Queries with Results:**

-- Query 1: List all Customer Assistance Staff with their associated Administrators

-- This query retrieves Customer Assistance Staff members and the Administrators they report to.

SQL Statement:

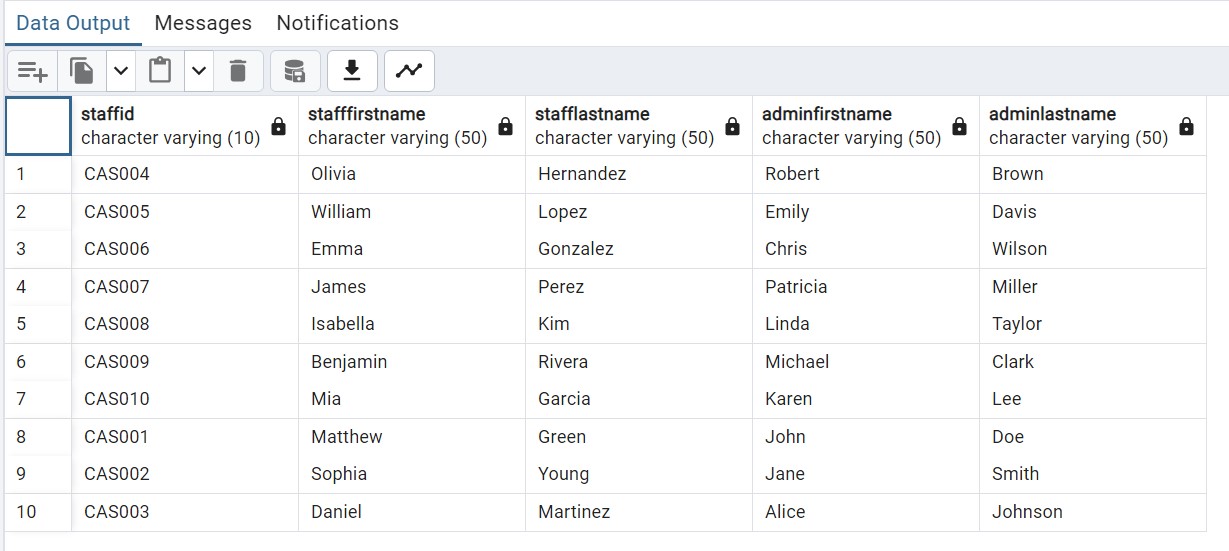
SELECT CAS.staffID, CAS.firstName AS StaffFirstName, CAS.lastName AS StaffLastName,

A.firstName AS AdminFirstName, A.lastName AS AdminLastName

FROM CustomerAssistanceStaff CAS

JOIN Administrator A ON CAS.adminID = A.adminID;

Results:



-- Query 2: Find the total number of trips completed by each customer

-- Aggregates the number of trips completed by each customer, grouping by customerID.

SQL Statement:

SELECT C.customerID, C.firstName, C.lastName, COUNT(T.tripID) AS TotalTrips

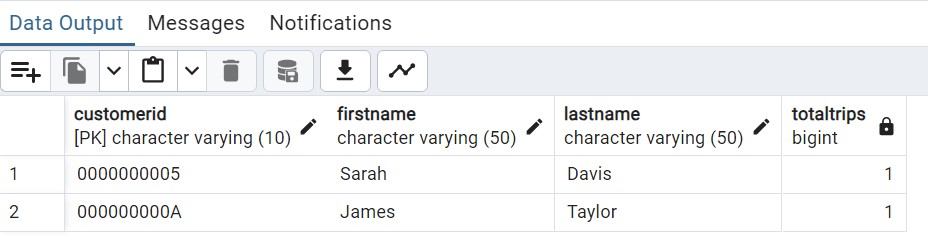
FROM Customer C

JOIN Trip T ON C.customerID = T.customerID WHERE T.status = 'completed'

GROUP BY C.customerID, C.firstName, C.lastName

HAVING COUNT(T.tripID) > 0;

Results:



-- Query 3: List all Maintenance Work Orders along with assigned staff and used products

-- Joins MaintenanceWorkOrder, AssignedTo, and Use tables to show details for each work order.

SQL Statement:

SELECT MWO.orderID, MWO.dateCreated, MS.firstName AS StaffFirstName, MP.productName

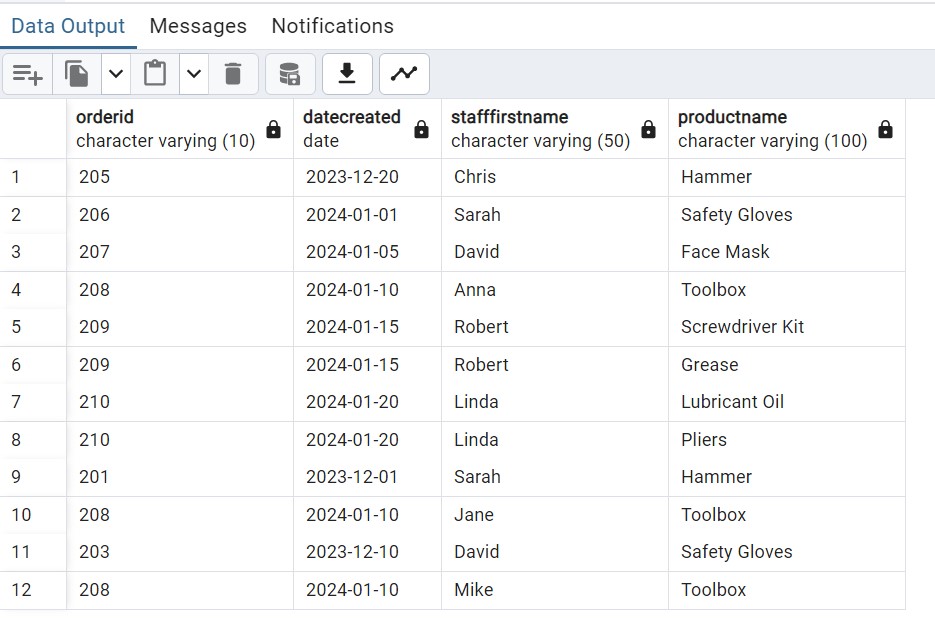
FROM MaintenanceWorkOrder MWO

JOIN AssignedTo AT ON MWO.orderID = AT.orderID

JOIN MaintenanceStaff MS ON AT.staffID = MS.staffID

JOIN Use U ON MWO.orderID = U.orderID

JOIN MaintenanceProducts MP ON U.productID = MP.productID; Results:



-- Query 4: Find drivers with insurance policies that expire within the next 6 months -- Uses a subquery to filter policies expiring within the next 6 months.

SQL Statement:

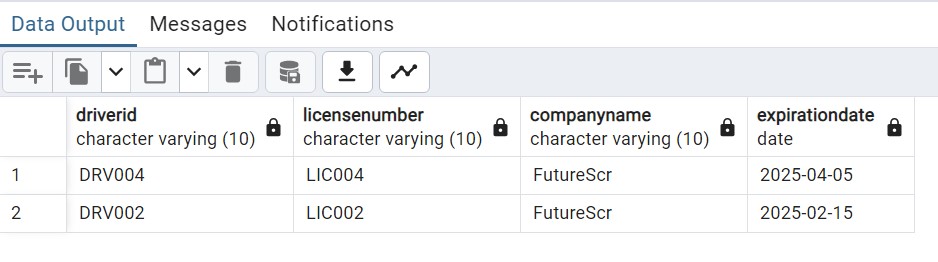
SELECT D.driverID, D.licenseNumber, DI.companyName, DI.expirationDate

FROM Driver D

JOIN DriverInsurance DI ON D.policyNumber = DI.policyNumber

WHERE DI.expirationDate < (CURRENT\_DATE + INTERVAL '6 months');

Results:



-- Query 5: Calculate average salary of employees grouped by their role (Administrator, Customer Assistance, Maintenance)

-- Uses multiple joins with GROUP BY to calculate the average salary per role.

SQL Statement:

SELECT 'Administrator' AS Role, AVG(A.salary) AS AverageSalary

FROM Administrator A

UNION

SELECT 'Customer Assistance' AS Role, AVG(CAS.salary) AS AverageSalary

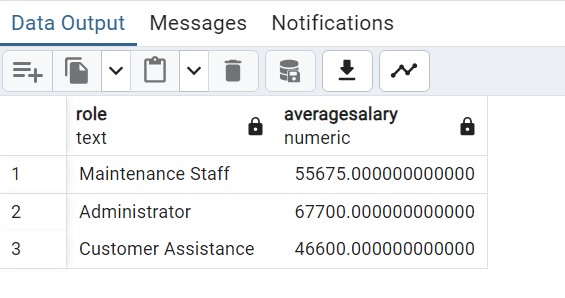
FROM CustomerAssistanceStaff CAS

UNION

SELECT 'Maintenance Staff' AS Role, AVG(MS.salary) AS AverageSalary

FROM MaintenanceStaff MS;

Results:



-- Query 6

-- Retrieve each customer's name, their payment card's expiration year, and the total distance of completed trips by the customer using aggregates(SUM) . This query also joins 4 tables to display the information needed. This also checked the trip is completed.

SQL Statement:

SELECT c.firstName AS customer\_first\_name, c.lastName AS customer\_last\_name, pi.ExpirationYear AS payment\_card\_expiration\_year,

SUM(t.distance) AS total\_completed\_trip\_distance

FROM Customer c

JOIN CustomerPaymentInfoLink cpil ON c.customerID = cpil.customerID

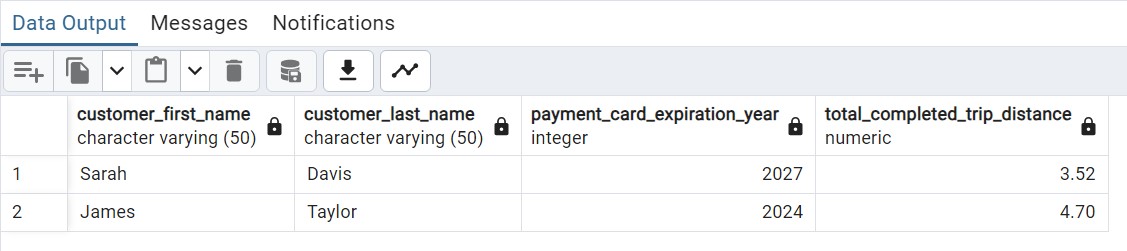
JOIN PaymentInfo pi ON cpil.cardNumber = pi.CardNumber

JOIN Trip t ON c.customerID = t.customerID

WHERE t.status = 'completed'

GROUP BY c.customerID, c.firstName, c.lastName, pi.ExpirationYear;

Results:



--Query 7

-- Retrieve each administrator’s name, the total number of customer assistance staff they manage, and the total salary paid to these staff members. This query join Administrator and CustomerAssistanceStaff tables while using aggregates(SUM) to calculate total salary and aggregates(COUNT) to count the staff that report to specific admin.

SQL Statement:

SELECT a.firstName AS admin\_first\_name, a.lastName AS admin\_last\_name, COUNT(cas.staffID) AS staff\_count,

SUM(cas.salary) AS total\_staff\_salary

FROM Administrator a

JOIN CustomerAssistanceStaff cas ON a.adminID = cas.adminID

GROUP BY a.adminID, a.firstName, a.lastName;

Results:



--Query 8

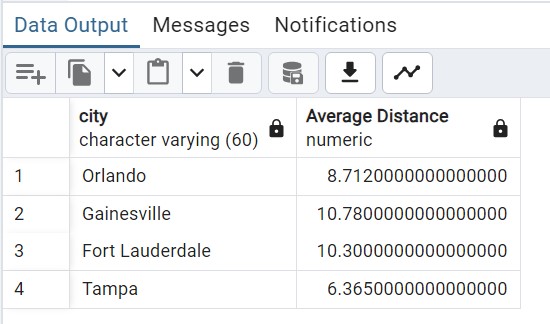
-- find the average distance of trips taken in each city as

-- indicated by the billing address of the card used for payment

SQL Statement:

select pc.city, AVG(distance) as "Average Distance" from Customer c join Trip t on c.customerID = t.customerID join PaymentInfo pi on t.cardNumber = pi.CardNumber join PostalCode pc on pi.postalCode = pc.postalCode group by pc.city;

Results:



--Query 9

-- Select all maintenance staff who have worked on taxi TX001

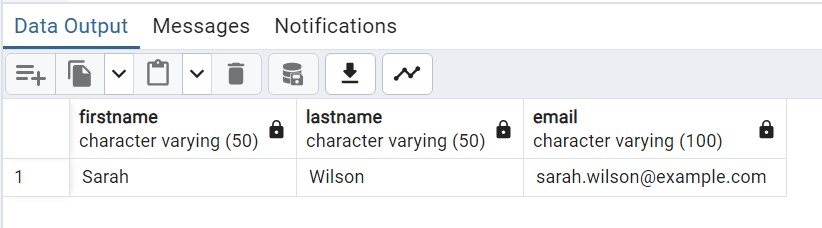
SQL Statement:

select firstName, lastName, email from MaintenanceStaff m where exists( select taxiID

from AssignedTo a join MaintenanceWorkOrder mwo using(orderID) join Taxi t using(TaxiID) where taxiID = 'TX001' and m.staffID = a.staffID

);

Results:



--Query 10

--Show the Policy number, company name, and monthly rate of all insurances from DriverInsurance that are above average

--Uses WHERE and subquery

SQL Statement:

SELECT driverID, i.policyNumber, i.companyName, i.monthlyRate FROM DriverInsurance i

JOIN Driver d ON i.policyNumber = d.policyNumber

WHERE monthlyRate >= (SELECT AVG(monthlyRate) FROM DriverInsurance);

Results:



--Query 11

--Find the last taxi work order of 2023, show the order id, taxi id, model, policy number, date order was created and fulfilled

--uses where and subquery

SQL Statement:

SELECT taxiID, model, policyNumber, orderID, dateCreated, dateFulfilled

FROM Taxi

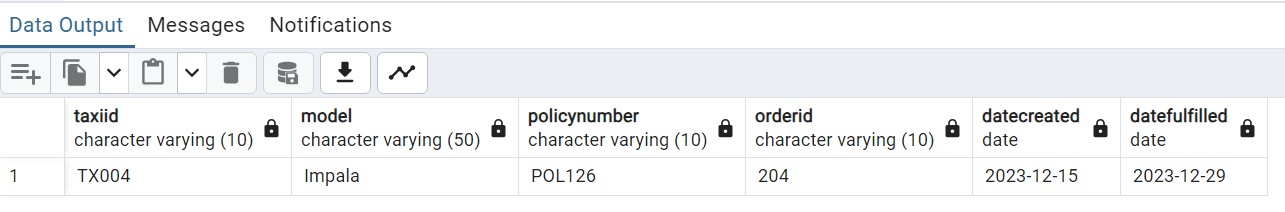
JOIN CarInsurance USING (taxiID)

JOIN MaintenanceWorkOrder USING (taxiID)

WHERE orderID = (SELECT MAX(orderID) FROM MaintenanceWorkOrder

WHERE dateFulfilled <= TO\_DATE('31-DEC-23', 'DD-MON-YY'));

Results:



# Conclusion

The development of the Taxi Management System

database provides an effective solution for managing taxi operations. This project included creating organized database structures, setting business rules, and writing SQL scripts for creating, inserting, updating, querying, and interacting with the data through constrain checks, views, triggers, and stored procedures.

The project focused on addressing practical operational needs through a well-structured database model while ensuring data accuracy with constraints like PRIMARY and FOREIGN keys, as well as CHECK and NOT NULL rules. It also showed how queries can provide useful information for decision-making, helping administrators, drivers, and customers.

This system is a reliable and flexible tool for tasks like managing trips, tracking maintenance, assigning drivers, and interacting with customers. By following the database design assignments correctly, the Taxi Management System ensures accurate data handling and is ready to support modern transportation needs.