## horizontal line



Car Fleet Management System

DBMS Project

**─**

Devanshi Dudhatra | 22BCP171

Heet Dobariya | 22BCP177

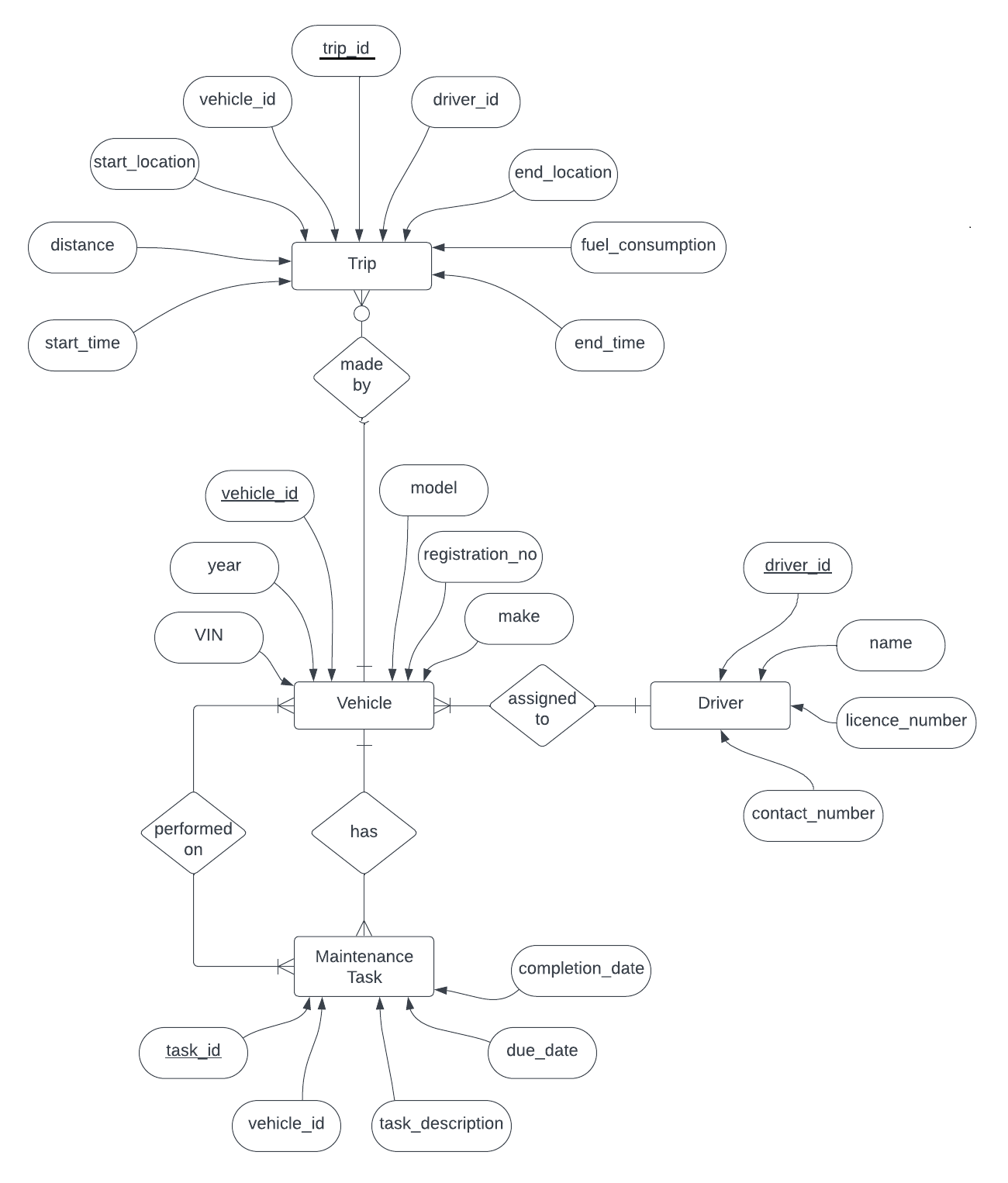
G5 | Division 3

# 

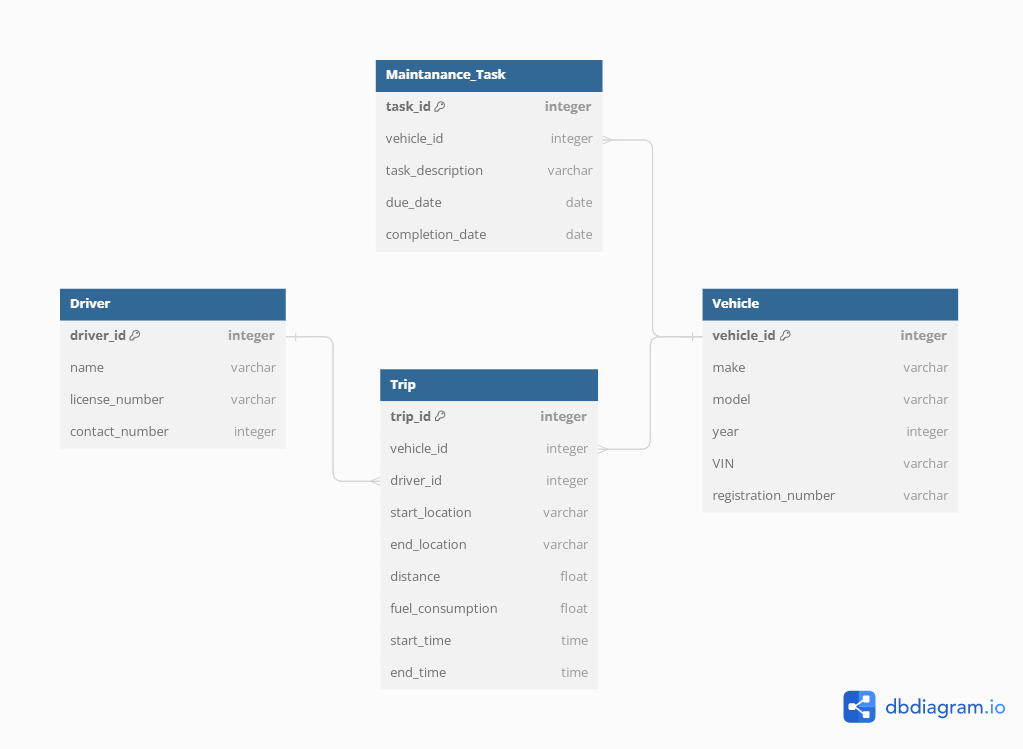
# Project Statement

* Project Overview: The project aims to develop a Car Fleet Management System that enables organizations to efficiently manage their vehicle fleet, including tracking vehicle usage, maintenance schedules, fuel consumption, and driver assignments.
* Use Cases (Applications):
  + Vehicle Management: Add, update, and retire vehicles from the fleet, track vehicle details such as make, model, year, VIN, and registration information.
  + Driver Management: Assign drivers to vehicles, track driver details such as name, license information, and contact details.
  + Maintenance Management: Schedule and track maintenance tasks for vehicles, record maintenance history, and generate alerts for upcoming services.
  + Trip Management: Track trips made by vehicles, including start and end locations, distance traveled, fuel consumption, and driver information.
* Objective: The objective is to create a comprehensive database management system that streamlines fleet operations, enhances vehicle utilization, and reduces maintenance costs.

# Entity Relationship Diagram



# Relational Model



# Normalization

* **For Vehicle (vehicle\_id, make, model, year, VIN, registration\_number):**
  + Functional Dependencies -

vehicle\_id → make, model, year, VIN, registration\_number

VIN → vehicle\_id

registration\_number → vehicle\_id

* Primary Key: vehicle\_id (correctly identifies each vehicle)
* No partial dependencies on the primary key. Therefore, it is already in 2NF.
* Now as it is in 2NF and it does not have any transitive dependencies i.e. no non-key attribute is dependent on any other non-key attribute. Therefore, it is also in 3NF.
* Highest Normal Form: 3NF
* **For Driver (driver\_id, name, license\_number, contact\_number):**
  + Functional Dependencies -

driver\_id → name, license\_number, contact\_number

license\_number → driver\_id

* Primary Key: driver\_id (correctly identifies each driver)
* No partial dependencies on the primary key. Therefore, it is already in 2NF.
* Now as it in 2NF and it does not have any transitive dependencies i.e. no non-key attribute is dependent on any other non-key attribute. Therefore , it is also in 3NF.
* Highest Normal Form : 3NF
* **For MaintenanceTask (task\_id, vehicle\_id, task\_description, due\_date, completion\_date):**
  + Functional Dependencies -

task\_id → vehicle\_id, task\_description, due\_date, completion\_date

vehicle\_id → task\_id

due\_date → task\_id

completion\_date → task\_id

* Primary Key: task\_id (uniquely identifies each task)
* Foreign Key: vehicle\_id references Vehicle(vehicle\_id) (establishes link with a vehicle)
* No partial dependencies on the primary key or foreign key. Therefore it is in 2NF.
* It is also in 3NF.
* Highest Normal Form : 3NF
* **For Trip (trip\_id, vehicle\_id, driver\_id, start\_location, end\_location, distance, fuel\_consumption, start\_time, end\_time):**
  + Functional Dependencies -

trip\_id → vehicle\_id, driver\_id, start\_location, end\_location, distance, fuel\_consumption, start\_time, end\_time

vehicle\_id → trip\_id, driver\_id, start\_location, end\_location, distance, fuel\_consumption, start\_time, end\_time

driver\_id → trip\_id, vehicle\_id, start\_location, end\_location, distance, fuel\_consumption, start\_time, end\_time

start\_location, end\_location, start\_time → trip\_id

end\_time → trip\_id, start\_time

* Primary Key: trip\_id (uniquely identifies each trip)
* Foreign Key: vehicle\_id references Vehicle(vehicle\_id) (establishes link with a vehicle)
* Foreign Key: driver\_id references Driver(driver\_id) (establishes link with a driver)
* No partial dependencies on the primary key or foreign keys.
* Therefore it is in 2NF.
* Also there are no transitive dependencies. Therefore it is in 3NF also.
* Highest Normal Form : 3NF

# SQL Implementation

**a. Create tables with necessary integrity constraints using SQL DDL statements.**

CREATE TABLE Vehicle (

vehicle\_id INT PRIMARY KEY,

make VARCHAR(50) NOT NULL,

model VARCHAR(50) NOT NULL,

year INT NOT NULL,

VIN VARCHAR(17) UNIQUE NOT NULL,

registration\_number VARCHAR(20) UNIQUE NOT NULL

);

CREATE TABLE Driver (

driver\_id INT PRIMARY KEY,

name VARCHAR(50) NOT NULL,

license\_number VARCHAR(20) UNIQUE NOT NULL,

contact\_number VARCHAR(20) NOT NULL

);

CREATE TABLE MaintenanceTask (

task\_id INT PRIMARY KEY,

vehicle\_id INT NOT NULL,

task\_description VARCHAR(200) NOT NULL,

due\_date DATE NOT NULL,

completion\_date DATE,

FOREIGN KEY (vehicle\_id) REFERENCES Vehicle(vehicle\_id)

);

CREATE TABLE Trip (

trip\_id INT PRIMARY KEY,

vehicle\_id INT NOT NULL,

driver\_id INT NOT NULL,

start\_location VARCHAR(100) NOT NULL,

end\_location VARCHAR(100) NOT NULL,

distance DECIMAL(10, 2) NOT NULL,

fuel\_consumption DECIMAL(10, 2) NOT NULL,

start\_time TIMESTAMP NOT NULL,

end\_time TIMESTAMP NOT NULL,

FOREIGN KEY (vehicle\_id) REFERENCES Vehicle(vehicle\_id),

FOREIGN KEY (driver\_id) REFERENCES Driver(driver\_id)

);

**b. Populate tables with relevant data.**

INSERT INTO Vehicle VALUES (1, 'Toyota', 'Camry', 2022, '1HGCM82633A004352', 'ABC123');

INSERT INTO Vehicle VALUES (2, 'Honda', 'Civic', 2020, 'JHMEH61600S215688', 'XYZ456');

INSERT INTO Vehicle VALUES (3, 'Ford', 'F-150', 2019, '1FTEW1EP9KFA25866', 'DEF789');

INSERT INTO Vehicle VALUES (4, 'Chevrolet', 'Tahoe', 2021, '1GNSKBKC0KR105746', 'GHI987');

INSERT INTO Vehicle VALUES (5, 'Nissan', 'Altima', 2018, '1N4AL3AP9JC167719', 'JKL654');

INSERT INTO Driver VALUES (1, 'John Doe', '123456', '9876543210');

INSERT INTO Driver VALUES (2, 'Jane Smith', '654321', '1234567890');

INSERT INTO Driver VALUES (3, 'David Johnson', '789012', '2345678901');

INSERT INTO Driver VALUES (4, 'Emily Davis', '890123', '3456789012');

INSERT INTO Driver VALUES (5, 'Michael Brown', '901234', '4567890123');

INSERT INTO Driver VALUES (6, 'Mukesh', '901553', '4565455323');

INSERT INTO MaintenanceTask VALUES (1, 1, 'Oil Change', '2024-03-20', NULL);

INSERT INTO MaintenanceTask VALUES (2, 2, 'Tire Rotation', '2024-03-22', NULL);

INSERT INTO MaintenanceTask VALUES (3, 3, 'Brake Inspection', '2024-03-25', NULL);

INSERT INTO MaintenanceTask VALUES (4, 4, 'Oil Change', '2024-03-28', NULL);

INSERT INTO MaintenanceTask VALUES (5, 5, 'Fluid Check', '2024-03-30', NULL);

INSERT INTO Trip VALUES (1, 1, 1, 'City A', 'City B', 100.5, 10.2, '2024-03-14 09:00:00', '2024-03-14 12:00:00');

INSERT INTO Trip VALUES (2, 2, 2, 'City C', 'City D', 120.8, 9.8, '2024-03-15 10:00:00', '2024-03-15 13:00:00');

INSERT INTO Trip VALUES (3, 3, 3, 'City E', 'City F', 80.3, 12.5, '2024-03-16 11:00:00', '2024-03-16 14:00:00');

INSERT INTO Trip VALUES (4, 4, 4, 'City G', 'City H', 150.2, 11.2, '2024-03-17 12:00:00', '2024-03-17 15:00:00');

INSERT INTO Trip VALUES (5, 5, 5, 'City I', 'City J', 200.5, 10.0, '2024-03-18 13:00:00', '2024-03-18 16:00:00');

**c. Write various SQL queries for data retrieval related to the application.**

-- Retrieve all vehicles assigned to a specific driver.

SELECT \* FROM Vehicle WHERE vehicle\_id IN (SELECT vehicle\_id FROM Trip WHERE driver\_id = 1);

-- Retrieve maintenance tasks due for a specific vehicle.

SELECT \* FROM MaintenanceTask WHERE vehicle\_id = 1 AND due\_date <= CURRENT\_DATE;

-- Calculate total distance traveled by a specific vehicle.

SELECT SUM(distance) AS total\_distance FROM Trip WHERE vehicle\_id = 1;

-- Find average fuel consumption for all trips made by a specific vehicle.

SELECT AVG(fuel\_consumption) AS average\_fuel\_consumption FROM Trip WHERE vehicle\_id = 1;

# UI

import streamlit as st

import sqlite3

from PIL import Image

# Connect to SQLite database

def connect\_to\_db():

return sqlite3.connect("FleetManagement.db")

# Function to create tables

def create\_tables():

conn = connect\_to\_db()

cursor = conn.cursor()

cursor.execute("""

CREATE TABLE IF NOT EXISTS Vehicle (

vehicle\_id INTEGER PRIMARY KEY,

make TEXT NOT NULL,

model TEXT NOT NULL,

year INTEGER NOT NULL,

VIN TEXT UNIQUE NOT NULL,

registration\_number TEXT UNIQUE NOT NULL

)

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS Driver (

driver\_id INTEGER PRIMARY KEY,

name TEXT NOT NULL,

license\_number TEXT UNIQUE NOT NULL,

contact\_number TEXT NOT NULL

)

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS MaintenanceTask (

task\_id INTEGER PRIMARY KEY,

vehicle\_id INTEGER NOT NULL,

task\_description TEXT NOT NULL,

due\_date DATE NOT NULL,

completion\_date DATE,

FOREIGN KEY (vehicle\_id) REFERENCES Vehicle(vehicle\_id)

)

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS Trip (

trip\_id INTEGER PRIMARY KEY,

vehicle\_id INTEGER NOT NULL,

driver\_id INTEGER NOT NULL,

start\_location TEXT NOT NULL,

end\_location TEXT NOT NULL,

distance DECIMAL(10, 2) NOT NULL,

fuel\_consumption DECIMAL(10, 2) NOT NULL,

start\_time TIMESTAMP NOT NULL,

end\_time TIMESTAMP NOT NULL,

FOREIGN KEY (vehicle\_id) REFERENCES Vehicle(vehicle\_id),

FOREIGN KEY (driver\_id) REFERENCES Driver(driver\_id)

)

""")

conn.commit()

conn.close()

# Function to insert data into a table

def insert\_data(table\_name, data):

conn = connect\_to\_db()

cursor = conn.cursor()

columns = ', '.join(data.keys())

placeholders = ', '.join(['?'] \* len(data))

query = f"INSERT INTO {table\_name} ({columns}) VALUES ({placeholders})"

cursor.execute(query, list(data.values()))

conn.commit()

conn.close()

# Function to delete data from a table

def delete\_data(table\_name, condition):

conn = connect\_to\_db()

cursor = conn.cursor()

query = f"DELETE FROM {table\_name} WHERE {condition}"

cursor.execute(query)

conn.commit()

conn.close()

# Function to search data from a table

def search\_data(table\_name, search\_criteria):

conn = connect\_to\_db()

cursor = conn.cursor()

query = f"SELECT \* FROM {table\_name} WHERE {search\_criteria}"

cursor.execute(query)

data = cursor.fetchall()

conn.commit()

conn.close()

return data

# Streamlit UI

def main():

st.title("Car Fleet Management Database")

# Sidebar for selecting operation

operation = st.sidebar.selectbox("Select Operation", ("View ER Diagram","View Data", "Add Data", "Delete Data","Search Data",))

if operation == "View Data":

st.subheader("View Data")

table\_name = st.selectbox("Select Table", ("Vehicle", "Driver", "MaintenanceTask", "Trip"))

conn = connect\_to\_db()

cursor = conn.cursor()

if table\_name == "Vehicle":

cursor.execute("SELECT \* FROM Vehicle")

data = cursor.fetchall()

columns = [description[0] for description in cursor.description] # Fetch column names

st.table([columns] + data) # Concatenate column names with data and display

elif table\_name == "Driver":

cursor.execute("SELECT \* FROM Driver")

data = cursor.fetchall()

columns = [description[0] for description in cursor.description] # Fetch column names

st.table([columns] + data) # Concatenate column names with data and display

elif table\_name == "MaintenanceTask":

cursor.execute("SELECT \* FROM MaintenanceTask")

data = cursor.fetchall()

columns = [description[0] for description in cursor.description] # Fetch column names

st.table([columns] + data) # Concatenate column names with data and display

elif table\_name == "Trip":

cursor.execute("SELECT \* FROM Trip")

data = cursor.fetchall()

columns = [description[0] for description in cursor.description] # Fetch column names

st.table([columns] + data) # Concatenate column names with data and display

conn.close()

elif operation == "Add Data":

st.subheader("Add Data")

table\_name = st.selectbox("Select Table", ("Vehicle", "Driver", "MaintenanceTask", "Trip"))

if table\_name == "Vehicle":

st.subheader("Add Vehicle")

make = st.text\_input("Make")

model = st.text\_input("Model")

year = st.number\_input("Year", min\_value=1900, max\_value=2100)

VIN = st.text\_input("VIN")

registration\_number = st.text\_input("Registration Number")

if st.button("Add Vehicle"):

insert\_data("Vehicle", {

"make": make,

"model": model,

"year": year,

"VIN": VIN,

"registration\_number": registration\_number

})

st.success("Vehicle added successfully!")

elif table\_name == "Driver":

name = st.text\_input("Name")

license\_number = st.text\_input("License Number")

contact\_number = st.text\_input("Contact Number")

if st.button("Add Driver"):

insert\_data("Driver", {

"name": name,

"license\_number": license\_number,

"contact\_number": contact\_number

})

st.success("Driver added successfully!")

elif table\_name == "MaintenanceTask":

vehicle\_id = st.number\_input("Vehicle ID", min\_value=1)

task\_description = st.text\_input("Task Description")

due\_date = st.date\_input("Due Date")

if st.button("Add Maintenance Task"):

insert\_data("MaintenanceTask", {

"vehicle\_id": vehicle\_id,

"task\_description": task\_description,

"due\_date": due\_date

})

st.success("Maintenance Task added successfully!")

elif table\_name == "Trip":

vehicle\_id = st.number\_input("Vehicle ID", min\_value=1)

driver\_id = st.number\_input("Driver ID", min\_value=1)

start\_location = st.text\_input("Start Location")

end\_location = st.text\_input("End Location")

distance = st.number\_input("Distance")

fuel\_consumption = st.number\_input("Fuel Consumption")

start\_time = st.text\_input("Start Time")

end\_time = st.text\_input("End Time")

if st.button("Add Trip"):

insert\_data("Trip", {

"vehicle\_id": vehicle\_id,

"driver\_id": driver\_id,

"start\_location": start\_location,

"end\_location": end\_location,

"distance": distance,

"fuel\_consumption": fuel\_consumption,

"start\_time": start\_time,

"end\_time": end\_time

})

st.success("Trip added successfully!")

elif operation == "Delete Data":

st.subheader("Delete Data")

table\_name = st.selectbox("Select Table", ("Vehicle", "Driver", "MaintenanceTask", "Trip"))

conn = connect\_to\_db()

cursor = conn.cursor()

if table\_name == "Vehicle":

st.subheader("Delete Vehicle")

vehicle\_id = st.number\_input("Vehicle ID", min\_value=1)

if st.button("Delete Vehicle"):

delete\_data("Vehicle", f"vehicle\_id = {vehicle\_id}")

st.success("Vehicle deleted successfully!")

elif table\_name == "Driver":

st.subheader("Delete Driver")

name = st.text\_input("Name")

if st.button("Delete Driver"):

delete\_data("Driver", f"name = '{name}'")

st.success("Driver deleted successfully!")

elif table\_name == "MaintenanceTask":

st.subheader("Delete Maintenance Task")

vehicle\_id = st.number\_input("Vehicle ID", min\_value=1)

if st.button("Delete Maintenance Task"):

delete\_data("MaintenanceTask", f"vehicle\_id = {vehicle\_id}")

st.success("Task deleted successfully!")

elif table\_name == "Trip":

st.subheader("Delete Trip")

start\_location = st.text\_input("Start Location")

end\_location = st.text\_input("End Location")

if st.button("Delete Trip"):

delete\_data("Trip", f"start\_location = '{start\_location}' and end\_location = '{end\_location}'")

st.success("Trip deleted successfully!")

elif operation == "Search Data":

st.subheader("Search Data")

table\_name = st.selectbox("Select Table", ("Vehicle", "Driver", "MaintenanceTask", "Trip"))

conn = connect\_to\_db()

cursor = conn.cursor()

if table\_name == "Vehicle":

st.subheader("Search Vehicle")

vehicle\_id = st.number\_input("Vehicle ID")

model = st.text\_input("Model")

VIN = st.text\_input("VIN")

registration\_number = st.text\_input("Registration Number")

if st.button("Search Vehicle"):

data = search\_data("Vehicle", f"vehicle\_id = {vehicle\_id}")

model\_data = search\_data("Vehicle" , f"model = '{model}'")

vin\_data = search\_data("Vehicle" , f"VIN = '{VIN}'")

reg\_data = search\_data("Vehicle" , f"registration\_number = '{registration\_number}' ")

if data:

st.table(data)

elif model\_data:

st.table(model\_data)

elif vin\_data:

st.table(vin\_data)

elif reg\_data:

st.table(reg\_data)

else:

st.warning("Vehicle not found.")

elif table\_name == "Driver":

st.subheader("Search Driver")

driver\_id = st.number\_input("Driver ID")

name = st.text\_input("Name")

license\_number = st.text\_input("License Number")

contact\_number = st.text\_input("Contact Number")

if st.button("Search Driver"):

data = search\_data("Driver", f"driver\_id = {driver\_id}")

name\_data = search\_data("Driver" , f"name = '{name}' ")

lic\_data = search\_data("Driver" , f"license\_number = '{license\_number}'")

contact\_data = search\_data("Driver" , f"contact\_number = '{contact\_number}'")

if data:

st.table(data)

elif name\_data:

st.table(name\_data)

elif lic\_data:

st.table(lic\_data)

elif contact\_data:

st.table(contact\_data)

else:

st.warning("Driver not found.")

elif table\_name == "MaintenanceTask":

st.subheader("Search Maintenance Task")

vehicle\_id = st.number\_input("Vehicle ID")

due\_date = st.date\_input("Due Date")

if st.button("Search Maintenance Task"):

data = search\_data("MaintenanceTask", f"vehicle\_id = {vehicle\_id}")

date\_data = search\_data("MaintenanceTask" , f"due\_date = {due\_date}")

if data:

st.table(data)

elif date\_data:

st.table(date\_data)

else:

st.warning("Maintenance task not found.")

elif table\_name == "Trip":

st.subheader("Search Trip")

start\_location = st.text\_input("Start Location")

end\_location = st.text\_input("End Location")

if st.button("Search Trip"):

data = search\_data("Trip", f"start\_location = '{start\_location}'")

end\_data = search\_data("Trip", f"end\_location = '{end\_location}'")

if data:

st.table(data)

elif end\_data:

st.table(end\_data)

else:

st.warning("Trip not found.")

conn.close()

elif operation == "View ER Diagram":

img = Image.open("ER Diagram.png")

st.image(img, width=700)

if \_\_name\_\_ == "\_\_main\_\_":

# Set background color using set\_page\_config

st.set\_page\_config(layout="wide",page\_title ="Fleet Management App", page\_icon=":car:",

initial\_sidebar\_state="expanded")

create\_tables()

main()