10/11/2025

Full stack Development Assessment 1

Database Design & Implementation 35%

Harry Parker
UNIVERSITY OF LINCOLN

Full stack development Assessment 1

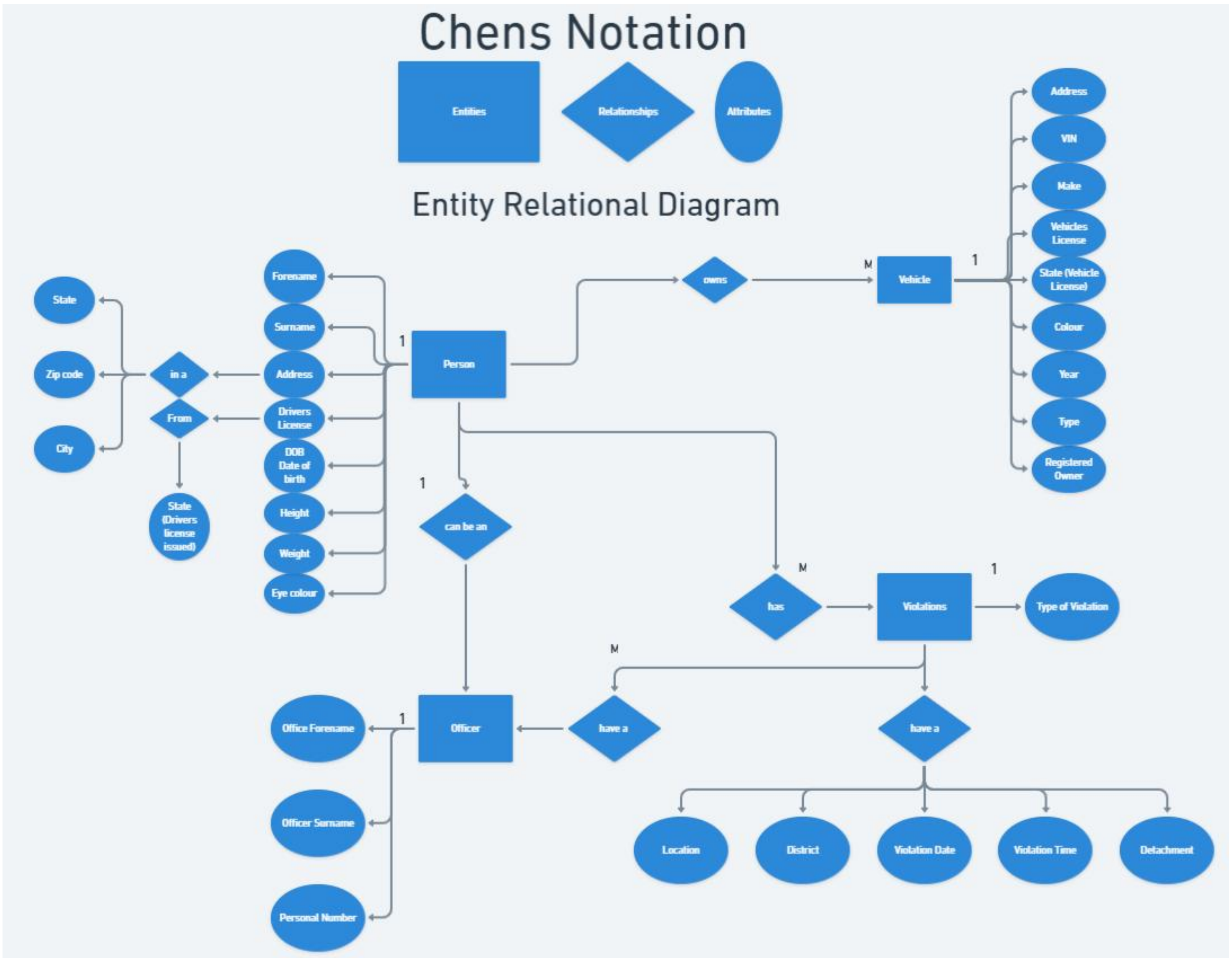
Table of Contents

Full stack Development Assessment 1 0
Full stack development Assessment 1 1
Table of Contents 1
Part 1 – Report 50% 2
Design 2
ERD 2
Description of ERD 2
Normalisation 3
Implementation 3
DDL 3
DCL 7
DDL 8
Reflections 13
References 13
Part 2 – MySQL Database (.sql) script 50% 13

Part 1 – Report 50%

Design

ERD



Description of ERD

On the above diagram, you can see an entity relation diagram. On there, you can see there are people, officers, violations, and cars, and you can see that different people can have many cars. One person can have many Violations. One violation can be set by multiple officers, one person can become one officer, and each individual entity can have one set of individual attributes (e.g. one person only has one set height or eye colour).

Normalisation

1st form

An example of where I used 1st form Normalisation, is when I separated details like Name into forename and surname. This helps to ensure the atomicity within the database and stops duplicate information. I also ensured that every table had uniquely identifiable records.

2nd form

An example of where I used 2nd form Normalisation, is when I split one large table into multiple entity-specific tables, assigned primary and foreign keys. An example of this being done, is where I tied Violation Date and time to ViolationsID making it so it is entirely linked to one entity.

3rd form

An example of where I used 3rd form Normalisation, is how originally, the State Drivers License issues was linked to the individual person, I countered this by creating the drivers license table which creates an ID and stores the people who have licenses. Another area where this was done, was the vehicle make, the year, weight, model could have all been repeated across a regular vehicle table, but instead of this constantly repeating data on a larger scale, I created a vehicletype table, to make it so it can be written once and just referenced instead.

Implementation

DDL

During the implementation of the schema, I used Data Definition language. Examples of this are below:

```
CREATE TABLE  tbl_DriversLicense( -- Creates table and names it

  DriversLicenseID int PRIMARY KEY NOT NULL, -- Adds column for Primary Key

  DriversLicenseOrigin varchar(50) -- Adds Column for Drivers License Origin

);


CREATE TABLE  tbl_Person( -- Creates table and names it

  PersonID int PRIMARY KEY NOT NULL AUTO_INCREMENT, -- Creates Primary Key for Person table

  DriversLicenseID int,

  FOREIGN KEY (DriversLicenseID) REFERENCES tbl_DriversLicense(DriversLicenseID), -- Creates Foreign Key to make it so drivers license and person tables can be joined to share information

  Forename varchar(50), -- set at 50 characters as most names do not exceed this amount
  Surname varchar(50), -- set at 50 characters as most names do not exceed this amount

  Address varchar(100), -- set at 100 characters as most addresses do not exceed this amount

  State varchar(100),  -- set at 100 characters as most States do not exceed this amount

  ZipCode varchar(5),  -- set at 5 characters as most ZipCodes do not exceed this amount

  City varchar(50),  -- set at 50 characters as most City names do not exceed this amount

  DOB varchar(10),   -- set at 10 characters as most DOB should not take more than that

  Height varchar(5),  -- set at 5 characters as Height should not exceed this amount

  Weight varchar(5),  -- set at 5 characters as Height should not exceed this amount

  EyeColour varchar(6),  -- set at 6 characters as Eye colour should not exceed this amount

  PhoneNum varchar(12),  -- set as varchar 12 as phone numbers only take set amounts of numbers, and numbers starting with 0 cause issues

  FOREIGN KEY (DriversLicenseID) REFERENCES tbl_DriversLicense(DriversLicenseID) -- Creates Foreign Key to make it so people can have Violations

);


CREATE TABLE  tbl_Officers( -- Creates and names table

  OfficerID int PRIMARY KEY NOT NULL AUTO_INCREMENT,

  PersonID int,

  FOREIGN KEY (PersonID) REFERENCES tbl_Person(PersonID), -- Sets foreign key

  ViolationsRecorded int

);


CREATE TABLE  tbl_ViolationTypes( -- Creates table and names it

  ViolationTypeID int PRIMARY KEY NOT NULL AUTO_INCREMENT, -- Adds column for Primary Key

  ViolationName varchar(100), -- Name of violation likely wont exceed 100 chars

  ViolationClass varchar(10) -- Penalty amount likely not higher than 25 chars

);
```

```
CREATE TABLE  tbl_Violations( -- Creates table and names it
   ViolationsID int PRIMARY KEY NOT NULL AUTO_INCREMENT, -- Adds column for Primary Key
   OfficerID int,
   FOREIGN KEY (OfficerID) REFERENCES tbl_Officers(OfficerID), -- Sets foreign key
   ViolationTypeID int,
   FOREIGN KEY (ViolationTypeID) REFERENCES tbl_ViolationTypes(ViolationTypeID), -- Sets foreign key
   Location varchar(100), -- Location of violation likely not more than 100 chars
   District varchar(100),
   ViolationDate varchar(10),
   ViolationTime varchar(8),
   ViolatorID int,
   FOREIGN KEY (ViolatorID) REFERENCES tbl_Person(PersonID) -- Sets foreign key
);


Create TABLE tbl_VehicleMake(
   VehicleMakeID int PRIMARY KEY NOT NULL AUTO_INCREMENT,
   VehicleName varchar(50), -- varchar 50 should not be exceeded in naming cars
   DriveType varchar(3), -- AWD, FWD, RWD
   VehicleManual bool, -- fale is automatic
   VehicleBrand varchar(50),
   CarWeight int,
   VehicleYear int
);


CREATE TABLE  tbl_Vehicles( -- Creates table and names it
   VIN int PRIMARY KEY NOT NULL AUTO_INCREMENT, -- Adds column for Primary Key
   VehicleLicense varchar(50), -- set at 50 characters as Vehicle license would likely not exceed this amount
   VehicleState varchar(50),-- set at 50 characters as Vehicle state would likely not exceed this amount
```

```
  VehicleColour varchar(35), -- set at 50 characters as longest vehicle colour is 35 chars at "British Racing Green Ultra Metallic"

  VehicleMakeID int,

  FOREIGN KEY (VehicleMakeID) REFERENCES tbl_VehicleMake(VehicleMakeID),

  VehicleAddress varchar(50), -- set as 50 as address likely isn't longer than 50

  VehicleOwner int,

  FOREIGN KEY (VehicleOwner) REFERENCES tbl_Person(PersonID) -- Sets foreign key

);
```

Now, I will talk about all of the different Create Table lines and explain why I used them the way I did:

1. Tbl_DriversLicense.
   - This is the Table for Drivers Licenses to be stored in. It needs to store the Drivers License ID and the origin of the License to make its place of origin be easily found.
   - I was considering putting all of the person information on here, but then It occurred to me that not all people would have licenses, so it would make more sense to put the information onto the Person table and have it link over to the DriversLicense table with the use of Foreign Keys.
   - The DriversLicense table has DriversLicenseID as the Primary Key. I learned how to create these from w3schools (w3schools, 2025).
2. Tbl_Person
   - This is the table for storing all information about the People. This includes information like name, address, state, height, weight, eye colour, DOB. This is all important information that can be used to differentiate people.
   - This has links to most of the other tables as it is the largest table that everything has been built around.
   - The Person table had primary and foreign Keys. The primary key auto incremented and was set as not null, I learned this from w3schools (W3Schools, 2025). I learned how to create the foreign Keys also from w3schools (W3Schools, 2025).
3. Tbl_Officers
   - This is the table for storing information about the officers. This only stores Officer IDs as it didn't need to store anything else. This is because with the use of queries, just knowing the ID of the officers allows us to know their names and all the information on tbl_Person, and we can also join with the Violations table to find out what violations they have set.
4. Tbl_ViolationTypes
   - This is the Violation Types table, this table exists to make it so instead of the same violation being written every single time someone commits it. A specific violation ID can just be referenced to say what violation was commit.
   - The creation of this table used the gov.uk site (Gov.Uk, 2025) to get accurate information for violations that could be added.
5. Tbl_Violations

- The Violations table stored the IDs from the people table to identify who commit a violation, it also used the IDs from the violation type table to say what violation was committed. As well as this, it also linked to the officer table to allow for queries to say which officer booked the violation.

6. Tbl_VehicleMake
    - The Vehicle make table was created so that on a larger scale, the database would not have the same car model being input over and over, and could instead just store the data in one vehicle make table that could be referenced for every vehicle stored in the vehicle table.
7. Tbl_Vehicles
    - The vehicle table was created to store information such as license plates, owners of cars and information similar to that. This links to the vehicle make table to make it so the information can be linked through just the use of a foreign key.

DCL

DCL is Data Control Language, this was used for creating accounts on the database, to make it so specific types of users would only have specific restricted access to ensure the correct usage of the schema. The code is pasted below.

The admin user has access to everything in the database, to allow for them to accurately maintain the database and keep it working exactly as expected.

The Officer user has access to the violations table, People table, violation types table and the vehicles table. This is to ensure that they can add anything they may need to, to their records. However, the officer is not able to delete any of the already made fields, so if there was an issue and one needed removing, this would need doing by the admin.

The Civilian has access to only see violations that have been committed. They are not able to change anything or see any more of the database as they would not need to.

```
-- ==========================================================

-- ROLEs

-- ==========================================================

-- Create Users login and passwords

CREATE USER 'Admin'@'%' IDENTIFIED BY 'AdminPassword';

CREATE USER 'Officer'@'%' IDENTIFIED BY 'OfficerPassword';

CREATE USER 'Civilian'@'%' IDENTIFIED BY 'CivilianPassword';


GRANT ALL PRIVILEGES ON FullstackDevelopment_Assignment1.* TO 'Admin'@'%'; -- Allows admins to create and delete and just do whatever they want

-- Only admins have access for vehicle types as officers would not have to create new vehicle types

GRANT SELECT, INSERT, UPDATE ON FullstackDevelopment_Assignment1.tbl_Violations TO 'Officer'@'%'; -- Allows officers to select, insert and update, but not delete

GRANT SELECT ON FullstackDevelopment_Assignment1.tbl_Person TO 'Officer'@'%';
GRANT SELECT ON FullstackDevelopment_Assignment1.tbl_ViolationTypes TO 'Officer'@'%';

GRANT SELECT ON FullstackDevelopment_Assignment1.tbl_Vehicles TO 'Officer'@'%';

-- Allows the Civilians to view violations but nothing else

GRANT SELECT ON FullstackDevelopment_Assignment1.tbl_Violations TO 'Civilian'@'%';


FLUSH PRIVILEGES;
```

DDL

DDL was used for creating at least 10 SQL queries. These are as follows:

Query 1: Create a new record for a correction notice

This query creates a new record for a correction notice.

```
INSERT INTO tbl_Violations(OfficerID, ViolationTypeID, Location, District, ViolationDate, ViolationTime, ViolatorID)

VALUES(2,5,"Oklahoma","County Sheriffs Office", "27/05/2025", "19:28:29",4);
```

Query 2: Update a correction notice: Previous query input incorrectly, needs updating

This query updates a previously input incorrectly notice that needs updating.

```
UPDATE tbl_Violations

SET Location = "Oklahoma Police Department"

WHERE OfficerID = 2

  AND ViolationTypeID =5

  AND Location = "Oklahoma"

  AND ViolatorID = 4

  AND ViolationDate = "27/05/2025"

  AND ViolationTime = "19:28:29";
```

Query 3: Delete an outdated or incorrect vehicle record

For this query, I needed an outdated or incorrect vehicle record, so I decided to make one entirely for this query. This can be seen below.

```
INSERT INTO tbl_Vehicles(VehicleLicense, VehicleState, VehicleColour, VehicleMakeID, VehicleAddress, VehicleOwner)

VALUES("ABD123", "Canada", "Green", 1, "123, springfield road", 1);
```

```
DELETE FROM tbl_Vehicles
WHERE VehicleLicense = "ABD123"
  AND VehicleState = "Canada"
  AND VehicleColour = "Green"
  AND VehicleMakeID = 1
  AND VehicleAddress = "123, springfield road"
  AND VehicleOwner = 1;
```

Query 4: List all violations with violator names, officer names, and violation type details

Here, I use the select to say I want the IDs, and names for officers and violators. These are taken from the violations table which is being referenced as v, this table has been joined with person table, Officer table, violator person table (which has been created to specifically hold the names for violators), and the violation types. This has then all been ordered by the Violation IDs.

```
SELECT
  v.ViolationsID,
  vt.ViolationName,
  violator.Forename AS ViolatorForename,
  violator.Surname AS ViolatorSurname,
  officerPerson.Forename AS OfficerForename,
  officerPerson.Surname AS OfficerSurname
FROM tbl_Violations v
JOIN tbl_Person violator
  ON v.ViolatorID = violator.PersonID
JOIN tbl_Officers o
  ON v.OfficerID = o.OfficerID
JOIN tbl_Person officerPerson
  ON o.PersonID = officerPerson.PersonID
JOIN tbl_ViolationTypes vt
  ON v.ViolationTypeID = vt.ViolationTypeID
ORDER BY v.ViolationsID;
```

Query 5: Retrieve all vehicles owned by people who have received violations

The Distinct key word makes it so if someone has more than one violation, there car will only appear once instead of over and over again, I learned this from W3Schools (W3Schools, 2025).

```
SELECT DISTINCT v.* -- Distinct makes it so if someone has more than one violation, there car will only appear once

FROM tbl_Vehicles v

JOIN tbl_Violations viol

  ON v.VehicleOwner = viol.ViolatorID;
```

Query 6: Show violations by Location and officer, grouped by district

In this query, we use the ON keyword, I learned this from HighTouch (HighTouch, 2025). This is used to "Join two or more tables to retrieve data from multiple sources and present it as a unified result set." (HighTouch, 2025). I also learned how to use CONCAT from w3schools (W3Schools, 2025) for this query. This was used to make it so we can display the name as a full name instead of just forename and then surname.

```
SELECT

  v.District,

  v.Location,

  CONCAT(op.Forename, ' ', op.Surname) AS OfficerName,

  COUNT(v.ViolationsID) AS NumberOfViolations

FROM tbl_Violations v

JOIN tbl_Officers o ON v.OfficerID = o.OfficerID

JOIN tbl_Person op ON o.PersonID = op.PersonID

GROUP BY v.District, v.Location, OfficerName
```

Query 7: Count of violations per officer over the past 25 Years

This query converts a string to a date to be able to compare the date of the violation to the current year and decide if it meets the 25 year requirement.

```
SELECT

  CONCAT(p.Forename, ' ', p.Surname) AS OfficerName,

  COUNT(v.ViolationsID) AS ViolationCount

FROM tbl_Violations v

JOIN tbl_Officers o ON v.OfficerID = o.OfficerID

JOIN tbl_Person p ON o.PersonID = p.PersonID
```

```
WHERE STR_TO_DATE(v.ViolationDate, '%d/%m/%Y') >= DATE_SUB(CURDATE(), INTERVAL 25
YEAR)

GROUP BY OfficerName

ORDER BY ViolationCount DESC;
```

Query 8: Count how many people under 50 have received violations

Similar to the last query, this one also converts the date string into an actual date for it to be compared against the current date of when it is run, this is done to find the violations committed in the last 50 years. In a larger database, this would likely be more beneficial to put to 21 years or even younger, but due to this database having such a small dataset, I set it to 50 to make it more clear.

```
SELECT

  COUNT(DISTINCT p.PersonID) AS Under21Violators

FROM tbl_Violations v

JOIN tbl_Person p ON v.ViolatorID = p.PersonID

WHERE TIMESTAMPDIFF(YEAR, STR_TO_DATE(p.DOB, '%Y-%m-%d'), CURDATE()) < 50;
```

Query 9: Count violations by location and type

This query counts all the violations recorded and counts them by location and organises them by type. This uses the COUNT keyword to count the violations, I learned this from W3Schools (W3Schools, 2025). It also uses the Group by and Order by keywords to be able to organise the data.

```
SELECT

  v.Location,

  vt.ViolationName,

  COUNT(v.ViolationsID) AS ViolationCount

FROM tbl_Violations v

JOIN tbl_ViolationTypes vt ON v.ViolationTypeID = vt.ViolationTypeID

GROUP BY v.Location, vt.ViolationName

ORDER BY v.Location, ViolationCount DESC;
```

Query 10: List all violators whose eye color is not blue AND height > '6' feet

This query selects data from the person table and the violations table, and links them together to be able to link the violations with the eye colour and height.

```
SELECT

  p.PersonID,
  p.Forename,
  p.Surname,
  p.Height,
  p.EyeColour
FROM tbl_Person p
JOIN tbl_Violations v ON p.PersonID = v.ViolatorID
WHERE p.EyeColour <> 'Blue'
 AND p.Height > 6;
```

Query 11: Retrieve vehicles that are manually driven and over a certain weight

This query grabs data from the vehicle make table, it joins with the regular vehicles tab and persons tab to be able to display the data for who owns the vehicle and how heavy the vehicle is, as well as knowing if the car is an automatic or a manual.

```
SELECT
  vm.VehicleName,
  vm.VehicleBrand,
  vm.VehicleYear,
  vm.CarWeight,
  vm.DriveType,
  v.VehicleColour,
  v.VehicleLicense,
  CONCAT(p.Forename, ' ', p.Surname) AS OwnerName
FROM tbl_VehicleMake vm
JOIN tbl_Vehicles v ON vm.VehicleMakeID = v.VehicleMakeID
JOIN tbl_Person p ON v.VehicleOwner = p.PersonID
WHERE vm.VehicleManual = 1
 AND vm.CarWeight > 1500
ORDER BY vm.CarWeight;
```

Reflections

In the future, if I were to do this again, I think I would spend more time working on the ERD, this is because after I finished working on it and moved onto the main schema, I noticed there were some areas where I had not done enough normalisation and had to go back and do it again.

As well as this, if I were to do this again, I would study how to populate tables through sql more, as when I had first started, I was doing an INSERT and VALUE line for every single piece of information, when I later on figured out I could use one INSERT and VALUE and have multiple sets of brackets to input multiple pieces of information at once. I learned how to populate the SQL through W3Schools (W3Schools, 2025).

References

Gov.Uk. (2025, 11 8). *Table of offences scheme*. Retrieved from Gov.Uk: https://www.cps.gov.uk/sites/default/files/documents/publications/annex_1a_table_of_offences_scheme_c.pdf

HighTouch. (2025, 11 8). *SQL Dictionary - SQL Join ON* . Retrieved from High Touch: https://hightouch.com/sql-dictionary/sql-join-on

W3Schools. (2025, 11 8). *SQL Autoincrement*. Retrieved from W3Schools: https://www.w3schools.com/sql/sql_autoincrement.asp

W3Schools. (2025, 11 8). *SQL COUNT*. Retrieved from W3Schools: https://www.w3schools.com/sql/sql_count.asp

W3Schools. (2025, 11 8). *SQL Distinct*. Retrieved from W3Schools: https://www.w3schools.com/sql/sql_distinct.asp

W3Schools. (2025, 11 8). *SQL Foreign Keys*. Retrieved from W3Schools: https://www.w3schools.com/sql/sql_foreignkey.asp

W3Schools. (2025, 11 8). *SQL Insert*. Retrieved from W3Schools: https://www.w3schools.com/sql/sql_insert.asp

w3schools. (2025, 11 8). *SQL Primary key*. Retrieved from W3Schools: https://www.w3schools.com/sql/sql_primarykey.asp

W3Schools. (2025, 11 8). *SQLserver- concat* . Retrieved from W3Schools: https://www.w3schools.com/sql/func_sqlserver_concat.asp

Part 2 – MySQL Database (.sql) script 50%

```sql
/*
Full stack Development Assignment 1 SQL code
Written By Harry Parker
Harry Parker : 29191718@students.lincoln.ac.uk
*/
```

```sql
-- ==========================================================
-- STEP 1: CREATE DATABASE AND TABLES
-- ==========================================================
CREATE DATABASE FullstackDevelopment_Assignment1; -- Creates the
Database/Schema that I will be working in
USE FullstackDevelopment_Assignment1; -- Tells the Software which Database I
will be working in

CREATE TABLE  tbl_DriversLicense( -- Creates table and names it
    DriversLicenseID int PRIMARY KEY NOT NULL, -- Adds column for Primary Key
    DriversLicenseOrigin varchar(50) -- Adds Column for Drivers License Origin
);

CREATE TABLE  tbl_Person( -- Creates table and names it
    PersonID int PRIMARY KEY    NOT NULL AUTO_INCREMENT, -- Creates Primary
Key for Person table
    DriversLicenseID int,
    FOREIGN KEY (DriversLicenseID) REFERENCES
tbl_DriversLicense(DriversLicenseID), -- Creates Foreign Key to make it so
drivers license and person tables can be joined to share information
    Forename varchar(50), -- set at 50 characters as most names do not exceed
this amount
    Surname varchar(50), -- set at 50 characters as most names do not exceed
this amount
    Address varchar(100), -- set at 100 characters as most addresses do not
exceed this amount
    State varchar(100),  -- set at 100 characters as most States do not exceed
this amount
    ZipCode varchar(5),  -- set at 5 characters as most ZipCodes do not exceed
this amount
    City varchar(50),  -- set at 50 characters as most City names do not
exceed this amount
    DOB varchar(10),   -- set at 10 characters as most DOB should not take
more than that
    Height varchar(5),  -- set at 5 characters as Height should not exceed
this amount
    Weight varchar(5),  -- set at 5 characters as Height should not exceed
this amount
    EyeColour varchar(6),  -- set at 6 characters as Eye colour should not
exceed this amount
    PhoneNum varchar(12),  -- set as varchar 12 as phone numbers only take set
amounts of numbers, and numbers starting with 0 cause issues
    FOREIGN KEY (DriversLicenseID) REFERENCES
tbl_DriversLicense(DriversLicenseID) -- Creates Foreign Key to make it so
people can have Violations
);
```

```
CREATE TABLE  tbl_Officers( -- Creates and names table
    OfficerID int PRIMARY KEY NOT NULL AUTO_INCREMENT,
    PersonID int,
    FOREIGN KEY (PersonID) REFERENCES tbl_Person(PersonID), -- Sets foreign
key
    ViolationsRecorded int
);

CREATE TABLE  tbl_ViolationTypes( -- Creates table and names it
    ViolationTypeID int PRIMARY KEY NOT NULL AUTO_INCREMENT, -- Adds column
for Primary Key
    ViolationName varchar(100), -- Name of violation likely wont exceed 100
chars
    ViolationClass varchar(10) -- Penalty amount likely not higher than 25
chars
);

CREATE TABLE  tbl_Violations( -- Creates table and names it
    ViolationsID int PRIMARY KEY NOT NULL AUTO_INCREMENT, -- Adds column for
Primary Key
    OfficerID int,
    FOREIGN KEY (OfficerID) REFERENCES tbl_Officers(OfficerID), -- Sets
foreign key
    ViolationTypeID int,
    FOREIGN KEY (ViolationTypeID) REFERENCES
tbl_ViolationTypes(ViolationTypeID), -- Sets foreign key
    Location varchar(100), -- Location of violation likely not more than 100
chars
    District varchar(100),
    ViolationDate varchar(10),
    ViolationTime varchar(8),
    ViolatorID int,
    FOREIGN KEY (ViolatorID) REFERENCES tbl_Person(PersonID) -- Sets foreign
key
);


Create TABLE tbl_VehicleMake(
    VehicleMakeID int PRIMARY KEY NOT NULL AUTO_INCREMENT,
    VehicleName varchar(50), -- varchar 50 should not be exceeded in naming
cars
    DriveType varchar(3), -- AWD, FWD, RWD
    VehicleManual bool, -- fale is automatic
    VehicleBrand varchar(50),
    CarWeight int,
    VehicleYear int
);

CREATE TABLE  tbl_Vehicles( -- Creates table and names it
    VIN int PRIMARY KEY NOT NULL AUTO_INCREMENT, -- Adds column for Primary
Key
    VehicleLicense varchar(50), -- set at 50 characters as Vehicle license
would likely not exceed this amount
    VehicleState varchar(50),-- set at 50 characters as Vehicle state would
likely not exceed this amount
    VehicleColour varchar(35), -- set at 50 characters as longest vehicle
colour is 35 chars at "British Racing Green Ultra Metallic"
    VehicleMakeID int,
    FOREIGN KEY (VehicleMakeID) REFERENCES tbl_VehicleMake(VehicleMakeID),
    VehicleAddress varchar(50), -- set as 50 as address likely isn't longer
than 50
    VehicleOwner int,
    FOREIGN KEY (VehicleOwner) REFERENCES tbl_Person(PersonID) -- Sets foreign
key
);


-- ==========================================================
-- STEP 2: POPULATE TABLES
-- ==========================================================


-- ==========================================================
-- Populating Violations Types
-- ==========================================================

INSERT INTO tbl_ViolationTypes(ViolationName, ViolationClass)
VALUES
("Abandonment of child under two","C"),
("Giving false statements to procure cremation","I"),
("Burglary (domestic)","E"),
("Burglary (non-domestic)","E"),
("Abduction of woman by force","J"),
("Forgery of driving documents","H"),
("Armed robbery","B");

-- ==========================================================
-- Populating Person Table
-- ==========================================================

INSERT INTO tbl_DriversLicense (DriversLicenseID, DriversLicenseOrigin)
VALUES
-- Person 1
(1, 'Mississippi State'),
-- Person 2
(2, 'Indiana'),
-- Person 3
(3, 'Texas'),
```

```sql
-- Person 4
(4, 'Kansas'),
-- Person 5
(5, 'Oklahoma');

INSERT INTO tbl_Person(DriversLicenseID, Forename, Surname, Address, State,
ZipCode, City, DOB, Height,  Weight, EyeColour, PhoneNum)
VALUES
-- Person 1
(1,"Ryan","Graham","30367 Alvarez Motorway","Mississippi State", "57843",
"West Laurenfort", "2001-11-14","6.1" ,"113.2", "Grey", "07968178644"),
-- Person 2
(2,"Rose","Davis","566 Peterson Skyway Apt. 948","Indiana State", "12346",
"Coxton", "1934-12-19","6.4" ,"52.2", "Blue", "07549552762"),
-- Person 3
(3,"Denise","Carter","723 Phillips Tunnel Suite 779","Texas State", "06724",
"North Steve", "1984-02-16","6.3" ,"92.4", "Brown", "07226683061"),
-- Person 4
(4,"John","Morgan","79850 Jermaine Burgs Apt. 950","Kansas State", "30767",
"Dianaview", "1961-09-19","6.6" ,"63.8", "Grey", "07990673772"),
-- Person 5
(5,"Michele","Sharp","04820 Nicole Points Apt. 978","Oklahoma State", "76305",
"Kristenberg", "1992-04-26","5.2" ,"56.8", "Blue", "07529054736");


-- ==========================================================
-- Populating Officers
-- ==========================================================

INSERT INTO tbl_Officers(PersonID)
VALUES
-- Officer 1
(1),
-- Officer 2
(5);


-- ==========================================================
-- Populating Violations
-- ==========================================================

INSERT INTO tbl_Violations(OfficerID, ViolationTypeID, Location, District,
ViolationDate, ViolationTime, ViolatorID)
VALUES
-- Officer 1
(1,7,"Texas","County Sheriffs Office", "12/03/1950", "16:42:59",2),
(1,1,"Oklahoma","Oklahoma City Police Department", "27/06/2003",
"11:27:48",3),
(1,5,"Kansas","Kansas City Police Department", "29/08/1998", "07:28:32",4),
(1,1,"Texas", "County Sheriffs Office", "03/02/1996", "14:56:12",3),
-- Officer 2
(2,4,"Indiana","Indianapolis Metropolitan Police Department (IMPD)",
"13/05/2000", "18:32:47",2);


-- =========================================================
-- Populating VehicleMakes
-- =========================================================
INSERT INTO tbl_VehicleMake(VehicleName ,DriveType, VehicleManual,
VehicleBrand,CarWeight, VehicleYear)
VALUES
("Toyota Corolla", "FWD", false, "Toyota", 1290, 2019),
("BMW M3", "RWD", true, "BMW", 1725, 2020),
("Ford F-150", "AWD", false, "Ford", 2040, 2021);

-- =========================================================
-- Populating Vehicles
-- =========================================================
INSERT INTO tbl_Vehicles(VehicleLicense, VehicleState, VehicleColour,
VehicleMakeID, VehicleAddress, VehicleOwner)
VALUES
("JTN-4821", "Texas", "Red", 1, "30367 Alvarez Motorway",1 ),
("7KZP310", "Indiana", "Green", 2, "566 Peterson Skyway Apt. 948",2 ),
("TXR-5M82", "Mississippi", "Orange", 3, "30367 Alvarez Motorway",5 );


-- =========================================================
-- ROLEs
-- =========================================================
-- Create Users login and passwords
CREATE USER 'Admin'@'%' IDENTIFIED BY 'AdminPassword';
CREATE USER 'Officer'@'%' IDENTIFIED BY 'OfficerPassword';
CREATE USER 'Civilian'@'%' IDENTIFIED BY 'CivilianPassword';

GRANT ALL PRIVILEGES ON FullstackDevelopment_Assignment1.* TO 'Admin'@'%'; --
Allows admins to create and delete and just do whatever they want
-- Only admins have access for vehicle types as officers would not have to
create new vehicle types
GRANT SELECT, INSERT, UPDATE ON
FullstackDevelopment_Assignment1.tbl_Violations TO 'Officer'@'%'; -- Allows
officers to select, insert and update, but not delete
GRANT SELECT ON FullstackDevelopment_Assignment1.tbl_Person TO 'Officer'@'%';
GRANT SELECT ON FullstackDevelopment_Assignment1.tbl_ViolationTypes TO
'Officer'@'%';
GRANT SELECT ON FullstackDevelopment_Assignment1.tbl_Vehicles TO
'Officer'@'%';
-- Allows the Civilians to view violations but nothing else
```

```sql
GRANT SELECT ON FullstackDevelopment_Assignment1.tbl_Violations TO
'Civilian'@'%';

FLUSH PRIVILEGES;

-- ==========================================================
-- Making Queries
-- ==========================================================
-- Query 1: Create a new record for a correction notice
INSERT INTO tbl_Violations(OfficerID, ViolationTypeID, Location, District,
ViolationDate, ViolationTime, ViolatorID)
VALUES(2,5,"Oklahoma","County Sheriffs Office", "27/05/2025", "19:28:29",4);

-- Query 2: Update a correction notice: Previous query input incorrectly,
needs updating
UPDATE tbl_Violations
SET Location = "Oklahoma Police Department"
WHERE OfficerID = 2
    AND ViolationTypeID =5
    AND Location = "Oklahoma"
    AND ViolatorID = 4
    AND ViolationDate = "27/05/2025"
    AND ViolationTime = "19:28:29";

-- Query 3: Delete an outdated or incorrect vehicle record
-- To start, I need an outdated or incorrect vehicle record, so I'll make one
now
INSERT INTO tbl_Vehicles(VehicleLicense, VehicleState, VehicleColour,
VehicleMakeID, VehicleAddress, VehicleOwner)
VALUES("ABD123", "Canada", "Green", 1, "123, springfield road", 1);

DELETE FROM tbl_Vehicles
WHERE VehicleLicense = "ABD123"
    AND VehicleState = "Canada"
    AND VehicleColour = "Green"
    AND VehicleMakeID = 1
    AND VehicleAddress = "123, springfield road"
    AND VehicleOwner = 1;

-- Query 4: List all violations with violator names, officer names, and
violation type details
SELECT
    v.ViolationsID,
    vt.ViolationName,
    violator.Forename AS ViolatorForename,
    violator.Surname AS ViolatorSurname,
    officerPerson.Forename AS OfficerForename,
    officerPerson.Surname AS OfficerSurname
```

```sql
FROM tbl_Violations v
JOIN tbl_Person violator
    ON v.ViolatorID = violator.PersonID
JOIN tbl_Officers o
    ON v.OfficerID = o.OfficerID
JOIN tbl_Person officerPerson
    ON o.PersonID = officerPerson.PersonID
JOIN tbl_ViolationTypes vt
    ON v.ViolationTypeID = vt.ViolationTypeID
ORDER BY v.ViolationsID;

-- Query 5: Retrieve all vehicles owned by people who have received violations
SELECT DISTINCT v.* -- Distinct makes it so if someone has more than one
violation, there car will only appear once
FROM tbl_Vehicles v
JOIN tbl_Violations viol
    ON v.VehicleOwner = viol.ViolatorID;

-- Query 6: Show violations by Location and officer, grouped by district

SELECT
    v.District,
    v.Location,
    CONCAT(op.Forename, ' ', op.Surname) AS OfficerName,
    COUNT(v.ViolationsID) AS NumberOfViolations
FROM tbl_Violations v
JOIN tbl_Officers o ON v.OfficerID = o.OfficerID
JOIN tbl_Person op ON o.PersonID = op.PersonID
GROUP BY v.District, v.Location, OfficerName
ORDER BY v.District, v.Location, OfficerName;

-- Query 7:Count of violations per officer over the past 25 Years
SELECT
    CONCAT(p.Forename, ' ', p.Surname) AS OfficerName,
    COUNT(v.ViolationsID) AS ViolationCount
FROM tbl_Violations v
JOIN tbl_Officers o ON v.OfficerID = o.OfficerID
JOIN tbl_Person p ON o.PersonID = p.PersonID
WHERE STR_TO_DATE(v.ViolationDate, '%d/%m/%Y') >= DATE_SUB(CURDATE(), INTERVAL
25 YEAR)
GROUP BY OfficerName
ORDER BY ViolationCount DESC;

-- Query 8: Count how many people under 50 have received violations

SELECT
    COUNT(DISTINCT p.PersonID) AS Under21Violators
FROM tbl_Violations v
```

```sql
JOIN tbl_Person p ON v.ViolatorID = p.PersonID
WHERE TIMESTAMPDIFF(YEAR, STR_TO_DATE(p.DOB, '%Y-%m-%d'), CURDATE()) < 50;

-- Query 9: Count violations by location and type
SELECT
    v.Location,
    vt.ViolationName,
    COUNT(v.ViolationsID) AS ViolationCount
FROM tbl_Violations v
JOIN tbl_ViolationTypes vt ON v.ViolationTypeID = vt.ViolationTypeID
GROUP BY v.Location, vt.ViolationName
ORDER BY v.Location, ViolationCount DESC;

-- Query 10:List all violators whose eye color is not blue AND height > ‘6’
feet

SELECT
    p.PersonID,
    p.Forename,
    p.Surname,
    p.Height,
    p.EyeColour
FROM tbl_Person p
JOIN tbl_Violations v ON p.PersonID = v.ViolatorID
WHERE p.EyeColour <> 'Blue'
  AND p.Height > 6;


-- Query 11: Retrieve vehicles that are manually driven and over a certain
weight
SELECT
    vm.VehicleName,
    vm.VehicleBrand,
    vm.VehicleYear,
    vm.CarWeight,
    vm.DriveType,
    v.VehicleColour,
    v.VehicleLicense,
    CONCAT(p.Forename, ' ', p.Surname) AS OwnerName
FROM tbl_VehicleMake vm
JOIN tbl_Vehicles v ON vm.VehicleMakeID = v.VehicleMakeID
JOIN tbl_Person p ON v.VehicleOwner = p.PersonID
WHERE vm.VehicleManual = 1
  AND vm.CarWeight > 1500
ORDER BY vm.CarWeight;
```