Database design rationale:

I used the coursework 2 database available on Moodle as a starting point and made several changes:

First, I created a new table to record the login details of police officers as well as whether or not they have administrator privileges. By default, new accounts will not be admin accounts, we set the auto incrementing UserID as the primary key, usernames should also be unique, and neither usernames nor passwords should be able to be set as NULL.

For one of the bonus features, I set up functionality for administrators to assign regular officers tasks to complete. These are stored in the Tasks table that I created along with the ID of the officer who is responsible for it. UserID also comes from the Users table so there is a foreign key relationship between the two tables.

At the bottom of this SQL file, I have made some modifications to the foreign key relationships. This is because I added the ability for officers to delete people, vehicles, and incidents from the database but there are default restrictions in place because of the foreign key relationships. If a person or vehicle is deleted, I have set the convention to ON DELETE CASCADE for the Ownership table because the ownership relationship should vanish when the vehicle or person is deleted. However, for the Incident table I have set the convention to ON DELETE SET NULL, because there is still useful incident information that should remain even if either the vehicle or person involved is deleted.

Website design rationale:

To design the website, I first split the requirements into four main categories: searching records, adding records, account management, and administrator functions. Therefore, I labelled the menu bar on my website to reflect these categories. Additionally, I decided to include a central home page in order to showcase some of the extra features that I added.

The menu bar, footer, and account verification take place on every page of the website, so I condensed these into separate functions in order to save lines of code (see functions.php).

In some cases, functionality can be split between displaying/validating forms and running SQL queries, so in these cases I split my code into two pages (e.g. addvehicle.php and addvehicleaction.php). This both simplifies my work and allows for cleaner navigation between the different parts of the program.

Note: most pages make use of HTML tables and forms and the various customisation tags. These elements are not mentioned below.

Page-by-page explanation

Page: functions.php

See comments written in the php file for more details

PHP: Includes three functions that will be called on most pages: verify() redirects users to the login page if they are not logged in and redirects regular users to the home page if they try to access administrator features, menu() adds the menu bar to the top of the page (which includes a police logo in the top left hyperlinking back to the home page, drop down menus that allow navigation to the different pages of the website, and an extra drop down menu for administrator features), footer() adds a footer to every page

Page: mvp.css

This is a CSS document provided by lecturers – here are a few adjustments I made:

- Changed the colour scheme of the template
- Changed the formatting of the <p> tag so that it is aligned in the centre of the page
- Added a new <h6> tag to display the central figure in the home screen dashboard
- Added a code segment at the bottom of this CSS file to set each page of the website as visible once rendering has finished. This prevents the Flash of Unstyled Content (see references for source).

Page: login.php

See comments written in the php file for more details

PHP: Checks whether there is any $_GET information in the URL and if there is, it will display a particular message depending on the value of the 'flag' in the URL

Javascript: Collects form input and checks whether this is empty and presents an alert if so

Page: loginaction.php

See comments written in the php file for more details

PHP: Either redirects the user on to home page or back to the login page depending on whether their login was successful

SQL: Counts the number of rows with username and password matching those provided in the login form

Page: home.php

See comments written in the php file for more details

PHP: Uses while loops to pipe the output from SQL queries into tables and bullet points, conditional statements are used to control which query is executed (used in the quick search functionality)

SQL: Performs the analytics for the three elements of the dashboard (selecting top three most common incidents, the top three offenders, and total incidents since 2017), queries are also used to search for any characters in any table (use in the quick search functionality), additional queries find the user ID of the police officer and then collect the task list for that particular officer (specified in the 'Tasks' table)

Page: changepassword.php

See comments written in the php file for more details

Javascript: function checks the length of new password and displays a pop up alert if it is too short (and does not submit the form in this case), this function also waits for your confirmation before submitting and changing your password

Page: changepasswordaction.php

See comments written in the php file for more details

PHP: Redirects back to changepassword.php once the page has finished running

SQL: Query is used to update the password for the row with the appropriate username

Searching, Editing & Deleting

Page: searchperson.php

See comments written in the php file for more details

PHP: Nested conditional statements are used to execute different parts of the code depending on the flag in the URL (controls whether all rows or a select number of rows from the People table are displayed, and displays an error message if there are no search results), while loops are used to pipe the SQL output into a table (HTML: hyperlinks in the table allow the user to edit or delete results from the table)

SQL: SELECT statements are used to either select all rows or rows where name or licence contains the characters specified in the search input

Page: editperson.php

See comments written in the php file for more details

PHP: Controls which fields we end up updating in the specified row from the People table

SQL: Finds the row we wish to edit and updates this row with the new information specified in the 'edit person' form

Page: deletepersonaction.php

See comments written in the php file for more details

PHP: Redirects back to searchperson.php after the row has been successfully deleted

SQL: Deletes the row the user has selected

Page: searchvehicle.php

See comments written in the php file for more details

PHP: Nested conditional statements are used to execute different parts of the code depending on the flag in the URL (controls whether all rows or a select number of rows from the Vehicle table are displayed, and displays an error message if there are no search results), while loops are used to pipe the SQL output into a table (HTML: hyperlinks in the table allow the user to edit or delete results from the table)

SQL: SELECT statements are used to either select all rows or rows where licence plate contains the characters specified in the search input

Page: editvehicle.php

See comments written in the php file for more details

PHP: Controls which fields we end up updating in the specified row from the Vehicle table

SQL: Displays the row we wish to edit and updates this row with the new information specified in the form (for updating the owner of a vehicle, there are extra queries to check if an ownership relation exists and either updates this relation or creates a new one depending on the answer)

Page: deletevehicleaction.php

See comments written in the php file for more details

PHP: Redirects back to searchvehicle.php after the row has been deleted

SQL: Deletes the row the user has selected

Page: searchreport.php

See comments written in the php file for more details

PHP: Nested conditional statements are used to execute different parts of the code depending on the flag in the URL (controls whether all rows or a select number of rows from the Incident table are displayed, and displays an error message if there are no search results), while loops are used to pipe the SQL output into a table (HTML: hyperlinks in the table allow the user to edit or delete results from the table)

SQL: SELECT statements to either select all rows or rows where the date or incident report contains the characters specified in the search input

Page: editreport.php

See comments written in the php file for more details

PHP: Controls which fields in the specified row we should be updating

SQL: Displays the row we wish to edit and updates this row with the new information specified in the form (for updating the person/vehicle ID fields, there is an extra query to convert the licence numbers the user enters into ID numbers)

Page: deletereportaction.php

See comments written in the php file for more details

PHP: Redirects back to searchreport.php after the row has been deleted

SQL: Deletes the row the user has selected

Adding Entries

Page: addperson.php

See comments written in the php file for more details

PHP: Sends a validation message that a new person has been added if we return to the page with a flag in the URL

Javascript: Validation function asking for user confirmation before new person is added

Page: addpersonaction.php

See comments written in the php file for more details

PHP: Uses conditional statements to control which SQL query we execute in order to add the new person. This is necessary since there are optional fields in the form which require modifications to the SQL query depending on the amount of information being inserted in the People table

SQL: Queries are used to insert the new person, with slight modifications depending on the optional fields in the form that are filled in

Page: addvehicle.php

See comments written in the php file for more details

PHP: Controls which buttons (e.g. 'search vehicle', 'back') are displayed depending on which form is being displayed, controls which form is being displayed depending on whether the officer wishes to enter an owner corresponding to the vehicle being added, provides different validation messages depending on whether only a vehicle or both a vehicle and an owner have been added, pipes the list of driving licences into the form to create a drop down list of licence numbers to choose from

SQL: Query is used to find a list of all non-null driving licences

Javascript: A separate validation function is used for each form. For the main vehicle addition form, the form checks if the officer wishes to add a corresponding owner and if not, a confirmation box asks whether the officer wishes to add the new vehicle. For the supplementary form, the validation function contains a set of logical conditions that prevent various errors in the officer's input (e.g. trying to add both a new and an existing owner of the vehicle, trying to use multiple means of person identification at the same time)

Page: addvehicleaction.php

See comments written in the php file for more details

PHP: Checks which optional fields were entered in the form and assigns session variables accordingly, controls which SQL queries run depending on which optional fields were entered, sends the user to various links depending on whether they need to fill out the supplementary form and when a confirmation message should be displayed

SQL: Various queries for inserting the new vehicle depending on which optional fields were filled out, converts from licence number to ID where necessary, inserts a new person where necessary and depending on which optional fields were filled in, records Vehicle ID too and uses this with person ID to insert a new ownership relation into the Ownership table

Page: addreport.php

See comments written in the php file for more details

PHP: Controls which form, which confirmation messages, and which buttons are displayed depending on the 'flag' value in the URL, while loops within each form help generate the drop down lists of driving licences and number plates

SQL: Queries within each form generate a list of person/vehicle licence numbers that are used to populate the drop-down lists

Javascript: A separate validation function is used for each form (four in total). The supplementary forms each contain a tailored set of logical conditions to prevent various errors in the officer's input (e.g. forgetting to add both a person and a vehicle). Confirmation boxes are also presented before the officer adds an incident

Page: addreportaction.php

See comments written in the php file for more details

PHP: Checks which optional fields were entered in the form and assigns session variables accordingly, controls which SQL queries run depending on which optional fields were entered, and depending on whether a vehicle and person are to be added to the incident, sends the user to various links to fill out the supplementary forms and to display confirmation messages

SQL: Converts from licence numbers to IDs where necessary, inserts new entries into the People and Vehicle tables where necessary, always inserts a new entry into the Incident table, there are a variety of queries whose structure depends on which optional fields are filled in, queries are also used to find the ID number of vehicles or people recently entered into the database

Administrator actions

Page: addfine.php

See comments written in the php file for more details

PHP: Presents a validation message that a fine has been added if we return to the page with a flag in the URL

Javascript: A validation function asks the user to confirm that they would like to add the new fine

Page: addfineaction.php

See comments written in the php file for more details

PHP: Redirects back to addfine.php with a flag in the URL

SQL: Query adds the new fine into the Fines table

Page: addtask.php

See comments written in the php file for more details

PHP: Presents a validation message that a task has been added if we return to the page with a flag in the URL

Javascript: A validation function asks the administrator to confirm that they would like to add the new task

Page: addtaskaction.php

See comments written in the php file for more details

PHP: Redirects back to addtask.php with a flag in the URL

SQL: Queries first find the police officer ID and then add the new task into the Tasks table using the located officer ID

Page: createaccount.php

See comments written in the php file for more details

PHP: Presents a validation message that an account has been added if we return to the page with a flag in the URL

Javascript: A validation function asks the administrator to confirm that they would like to add the new account and also makes sure that the password chosen is long enough

Page: createaccountaction.php

See comments written in the php file for more details

PHP: Redirects back to createaccount.php with a flag in the URL

SQL: Query adds the new account into the Users table

List of included files (excluding cover pages, manuals, etc.)

- addfine.php
- addfineaction.php
- addperson.php
- addpersonaction.php
- addreport.php
- addreportaction.php
- addtask.php
- addtaskaction.php
- addvehicle.php

- addvehicleaction.php
- changepassword.php
- changepasswordaction.php
- createaccount.php
- createaccountaction.php
- databasecode.sql
- deletepersonaction.php
- deletereportaction.php
- deletetaskaction
- deletevehicleaction.php
- editperson.php
- editreport.php
- editvehicle.php
- functions.php
- home.php
- login.php
- loginaction.php
- mvp.css
- police_logo.png
- searchperson.php
- searchreport.php
- searchvehicle.php