Table of Contents

[Login: 2](#_Toc479157980)

[User Related Forms 3](#_Toc479157981)

[Homepage Form 3](#_Toc479157982)

[Homepage Handle Form 4](#_Toc479157983)

[Reservation Handle Form 6](#_Toc479157984)

[Homepage Query Locations 7](#_Toc479157985)

[Member Rental History 7](#_Toc479157986)

[Pickup/Dropoff Form – pdform php file 8](#_Toc479157987)

[Pickup Dropoff Form – pdhandle php file 10](#_Toc479157988)

[Admin 13](#_Toc479157989)

[Admin Homepage Form 13](#_Toc479157990)

[Monthly Invoice Handle Form 16](#_Toc479157991)

[Add Car Form 16](#_Toc479157992)

[Add Car Handle Form 16](#_Toc479157993)

[Car Rental History Handle Form 18](#_Toc479157994)

[Available Cars at Location Form 18](#_Toc479157995)

[Damaged or Need to be Repaired Cars Handle 18](#_Toc479157996)

[Rental Comment Handle Form 19](#_Toc479157997)

[Rental Comment Obj Form 19](#_Toc479157998)

[Insert Reply Comment Form 19](#_Toc479157999)

[Cars Distance Since Maintenance Handle 20](#_Toc479158000)

[Logout 21](#_Toc479158001)

[Appendix 22](#_Toc479158002)

[SQL Dumps of All Tables Preloaded with the KTCS\_DB script 22](#_Toc479158003)

[Admin Table 22](#_Toc479158004)

[Car Table 22](#_Toc479158005)

[Car\_Maintenance\_History Table 23](#_Toc479158006)

[Car\_Rental\_History Table 23](#_Toc479158007)

[KTCS\_Member Table 23](#_Toc479158008)

[Parking\_Location Table 24](#_Toc479158009)

[Payment\_History Table 24](#_Toc479158010)

[Rental\_Comment Table 24](#_Toc479158011)

[Reservations Table 25](#_Toc479158012)

# Login:

The Login statechart internal transitions consists of the following SQL queries to validate the user/admin.

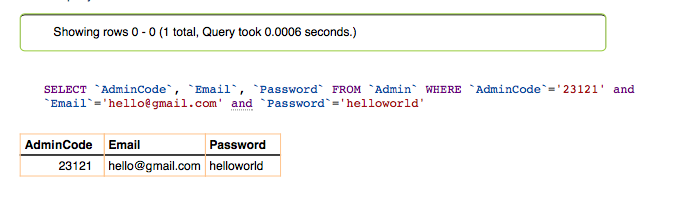
Please note that in the php code files created the needed values for the queries are dynamic and are variables. When running all the following SQL queries the value for the given variable at that point in time was set into the SQL query.

The following queries try to validate the user email ‘[testuser@gmail.com](mailto:testuser@gmail.com)’ with the password ‘testing’.



When verifying the user, the users text password entered in the form is hashed and is compared to the hash password in the database seen above. If both email address and the hashed password match the values stored in the database, the user will successfully be logged in.

In terms of verifying an admin the following SQL query was used. The values seen in SQL statement below were the values the user entered into the Login form.

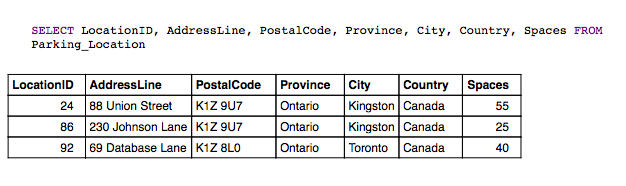


# User Related Forms

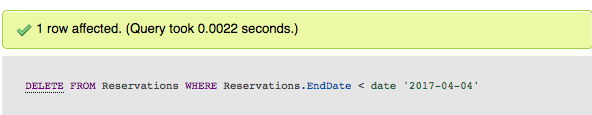
## Homepage Form

The user homepage form has the following SQL queries executed within it.

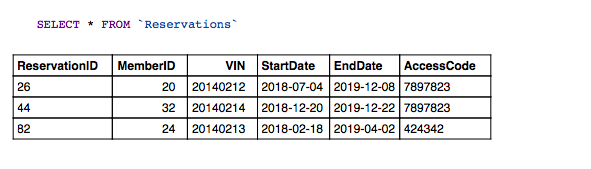
Firstly, within the homepage users will be able to reserve a car. In order to begin the reservation process users, specify a start and end date for the reservation and choose from one of the Parking location spots from a dropdown list. This parking location dropdown list is dynamic and retrieves the parking locations from the database with the following query:



In addition, the user homepage executes another query every time the user accesses the homepage/reloads it. This query is responsible for consistently checking and deleting reservations that have already happened ie. Reservation object end date is less than today’s date. Reservation objects should be deleted as they complete as the reservation object is converted to a car rental history object upon its completion. The following SQL query will delete any and all old reservation objects:



The DELETE query does not return any results, but as seen above it did indeed delete one row whose reservation end date was less than today’s date (April 4th 2017). Doing a current dump of the Reservations table (seen below) it is evident that there is no reservation whose end date is less than today’s date.

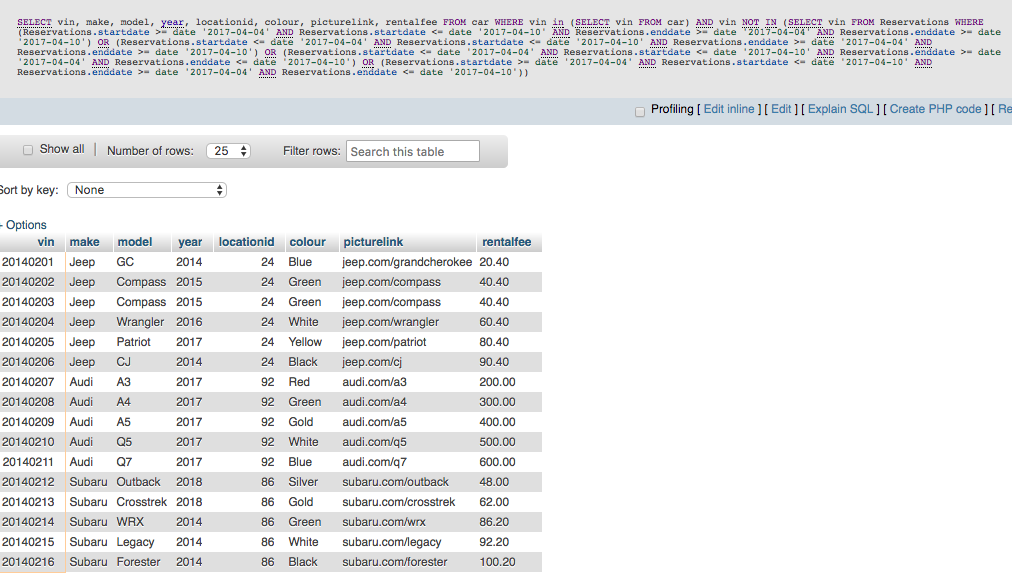


## Homepage Handle Form

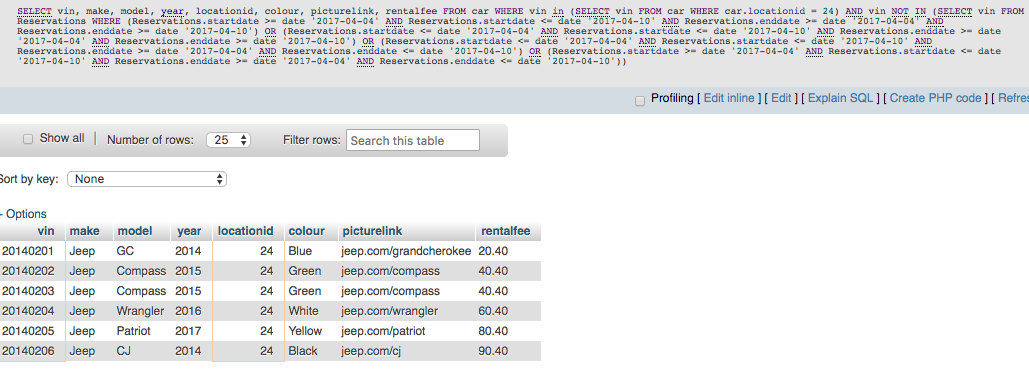
The homepagehandle php file is called when the user clicks submit in the homepageform.

The homepagehandle form displays all the cars that are available to reserve for the specified dates. The user specified dates are passed from the homepageform (where they are entered by the user) to this form to be used for the following queries:

**Please note the following queries are for the user entered start date of 2017-04-04 and end date of 2017-04-10.**

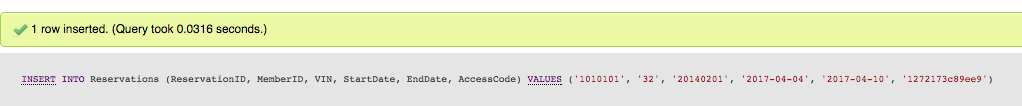
If the user selected the ‘All’ locations in the homepage the following query will be performed and the following cars will be displayed:

If the user selected a specific location such as ’88 union street’ which has a location ID of 24, the following query will be performed:

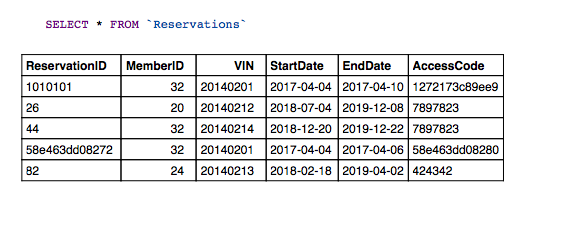


Overall, the difference between choosing the ‘All’ locations or a specific location is whether the query looks at the location ID or if it instead selects all location ID’s (in the case of ‘All’).

## Reservation Handle Form

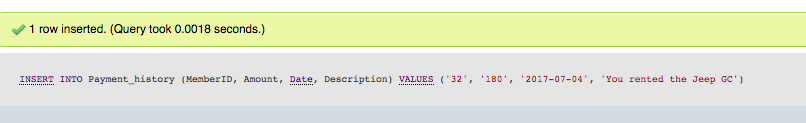
The reservationhandle php file will get the given car info (such as VIN, make, model, year, etc..) from the homepagehandle form as every car shown in the homepagehandle form has an associated ‘Reserve a Car’ button. Upon retrieving this information, the following queries are performed to reserve the car:

The above query will insert the new Reservation into the Reservation table. To confirm that the query is correct the dump of the Reservation table can be seen below.

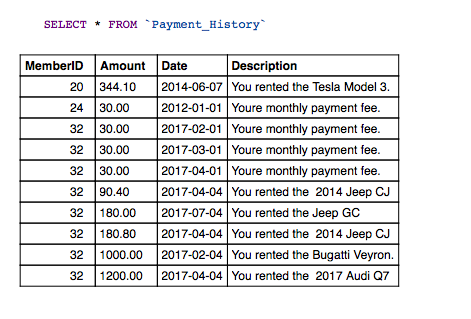


From the above dump, it is evident that the ‘20140201’ VIN reservation object was inserted.

In addition, the reservationhandle form will also create a payment object with the following query:



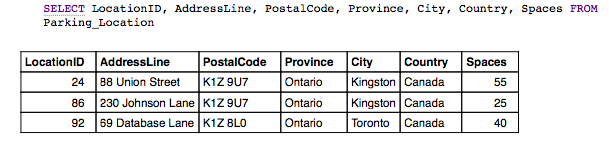
Performing a dump on the Payment\_History table gives the following results:



It is evident that the new Payment object was inserted successfully.

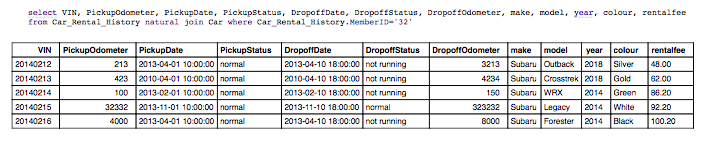
## Homepage Query Locations

The homepageQueryLocations php file displays all the Parking\_Location objects and their information to the user. The SQL query performed is seen below.



## Member Rental History

The following form presents all of the users previously completed rentals and the information about each of them. The form performs the following SQL query:



## Pickup/Dropoff Form – pdform php file

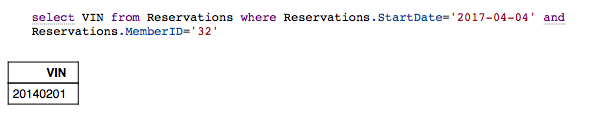
The following form is an extremely dynamic form. It can take on 4 different internal states. The first internal state it can take on is if there are no reservations whose start date is today’s date, it will simply display text informing the user there is no pickup form to complete. To check if any reservations become active today, the SQL statement used is:

select VIN

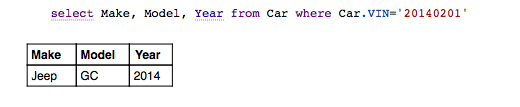
from Reservations

where Reservations.StartDate='$currentDate' and Reservations.MemberID='$memberID'

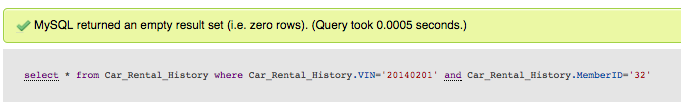
Please note that $currentDate and $memberID are defined php variable in pdform.php. Running this query with current date of 2017-04-04 and memberID of 32 gives the following results:



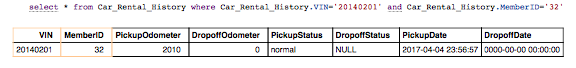
If the result returns at least 1 row or more (in this case it does) then there is a reservation object with the following VIN seen above starting today. If this is the case the following query will be ran to collect the car information for the VIN from the previous query.



Now the following query is executed to check whether a Car\_Rental\_History object already exists for the given VIN or not. The purpose of this is to determine if the form should dynamically change to its other internal states which is show the pickup form values or show the dropoff form values.



As can be seen above, if no rows are returned then this means the Car\_Rental\_History object does not exist for the current car, so the pickup state form will be shown asking the user to enter the current odometer and status of car values. Upon submitting the pickup values and these values being handled by the pdhandle php file a Car\_Rental\_History object will be created for the given car. It will only contain the pickupOdometer and pickupStatus values the user entered as well as the pickup timestamp. All the dropoff values will initially be set to NULL (if user has filled in the pickup form). Now, when user goes to the pickup/dropoff form again (pdform.php) it will once again check the first query seen above. This will still return a result greater than or equal to 1, so the pdform will now re-try the above query.



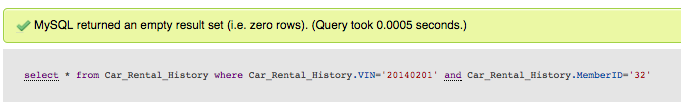
As can be seen from above, the SQL query returned a valid result. This means that the pickup values were set, but the dropoff values were not set. The pdform will now analyze the contents of the dropoffOdometer column. If it finds that the dropoffOdometer has not been set (ie. Its 0 or null) it will know to show the dropoff form and not the pickup form. Upon showing the drop off form the user will be able to enter the dropoffOdometer and dropoffStatus. The dropoffDate timestamp will be generated by the scripts. In addition, in the dropoff form the user will be able to provide feedback on the rental by entering the rating and optionally a comment for the rental. These values are passed onto the pdhandle php form.

The final internal state in the pdform is the already completed rental text shown to the user. This text will be show to the user if all the above query’s column values are filled with valid results (ie. Not null or 0). This implies that the user has completed both the pickup and dropoff aspects of the form and so the form will not allow the user to re-complete the form.

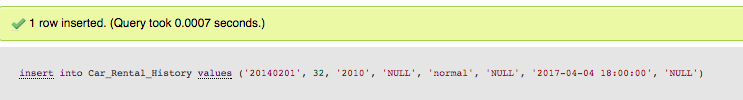
## Pickup Dropoff Form – pdhandle php file

The pdform file passes the user entered input to the given internal state (pickup or dropoff form values) to the pdhandle file upon the user pressing the submit button in pdform. The pdhandle will perform the following queries:

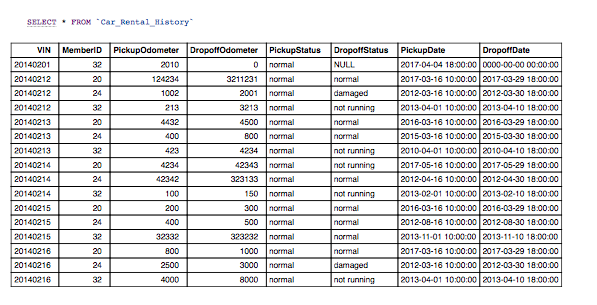
Firstly, the pdhandle file will check to see if there currently exist any objects of Car\_Rental\_History for the given VIN and memberID. This query is essentially the same as that of the pdform. Executing this query initially produces the following results:



Since the SQL query did not return any results, the pdhandle will create a Car\_Rental\_History object with the user entered values (for the pickup part of the form) from the pdform file. This can be seen below:

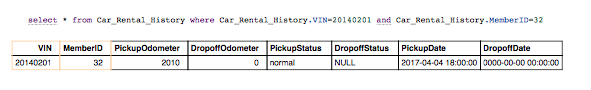


To confirm this worked the Car\_Rental\_History dump is seen below.

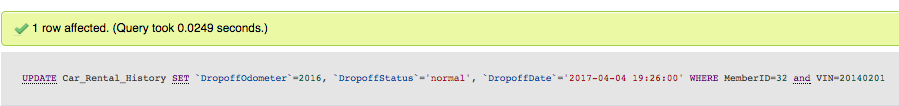


From the results, above, it can be confirmed that the new object was added to the table (the object is seen as the first value in the dump).

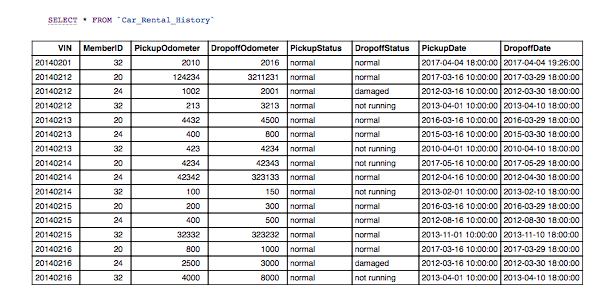
Since the object has now been created and populated with the pickup values the next time pdform calls pdhandle it will pass in the dropoff values. The same first query will be executed in pdhandle to determine if the Car\_Renal\_History objects exists for the given VIN and memberID. This time around it will indeed find the object with the pickup values set as seen below.



Since the pickup values are set, the pdhandle will check to see whether the dropoffOdometer value is set. In this case it is not and therefore the pdhandle will invoke the following query to update this object with the dropoff values supplied by the user in pdform.

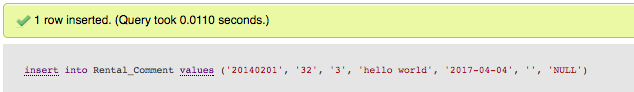


To confirm whether this query worked the table dump can be seen below.

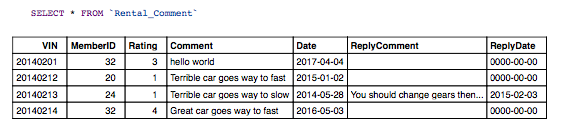


As seen from the table dump above, it is evident that the object was updated with the dropoff values.

In addition to updating the dropoff values in the correct Car\_Rental\_History object the pdhandle will also take the user entered rating and rental comment values (user supplied these values in pdform) and create a Rental\_Comment object. The following SQL demonstrates this.



Doing a table dump produces the following results.



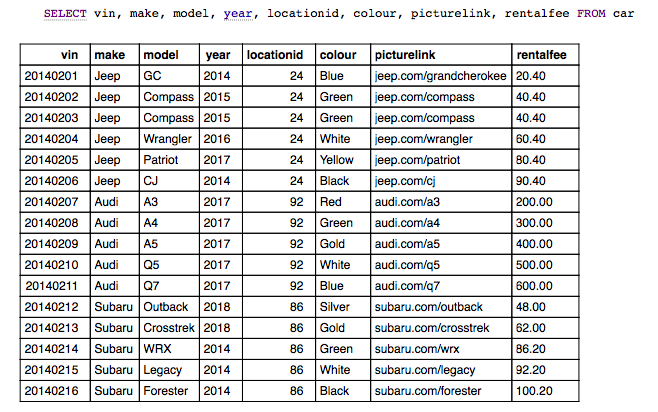
It is evident from above that the new Rental\_Comment object has been inserted. In terms of the ReplyComment by default this is nothing as the admin has not replied to the comment. If the Admin wishes to reply to the users comment in the Admin portal this table will be updated and the change will be reflected.

# Admin

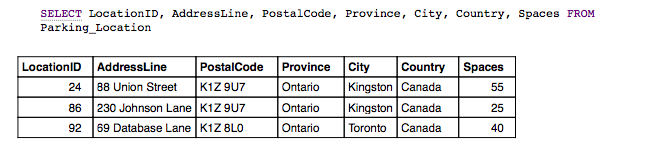
## Admin Homepage Form

The admin homepage form (adminHomePage.php) contains various SQL queries. Firstly, within this form there is a dropdown allowing an admin to choose a specific user for the monthly invoice section. The dropdown consisting of all the KTCS\_Members performs the following query.

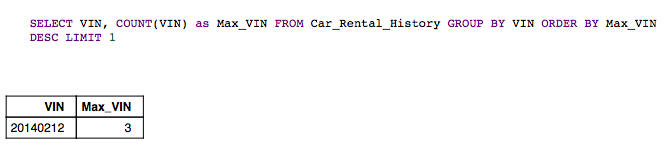
Another query the admin homepage performs is to dynamically load the dropdown list with all the cars in the database for the ‘View Rental History for Cars’ section. The query is seen below.



For the ‘Available Cars at Location’ section of the admin homepage the following query is performed to retrieve the dynamic parking locations dropdown list.

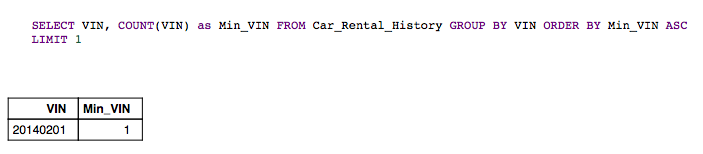


For the ‘View Cars with Most/Minimum Rentals’ section the following set of queries is ran:



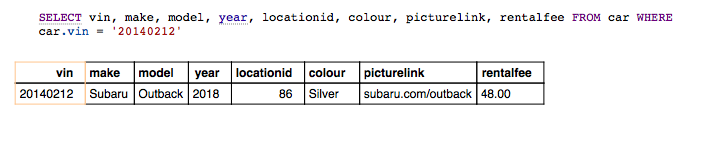
The purpose of the above query is to count the occurrences of each VIN and order in ascending. LIMIT 1 is used to only select the VIN that contains the most occurrences as the max rental is desired.

Similarly, to find the VIN that contains the minimum number of occurrences the following query was ran.

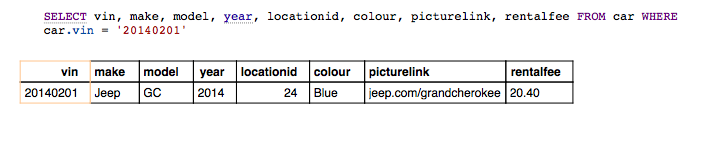


Now that the VIN’s for the max and min rental were found the information for the given VIN’s is found and shown to the user using the following queries:

Max Rental – VIN found above to be 20140212

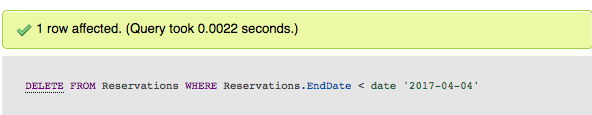


Min Rental – VIN found above to be 20140201

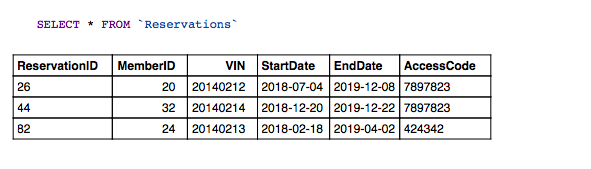


As for the ‘View Rental Comments’ section of the admin homepage the Members dynamic dropdown list is populated through the following query.

Finally, similarly to the user homepage the last query the admin homepage runs is to delete any Reservation objects whose end dates are less than the current date. The following query accomplishes this.

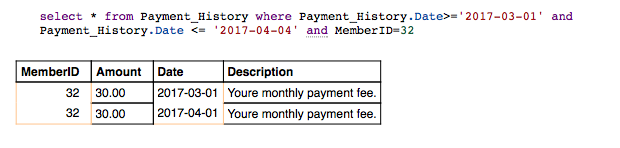


The DELETE query does not return any results, but as seen above it did indeed delete one row whose reservation end date was less than today’s date (April 4th 2017). Doing a current dump of the Reservations table (seen below) it is evident that there is no reservation whose end date is less than today’s date.



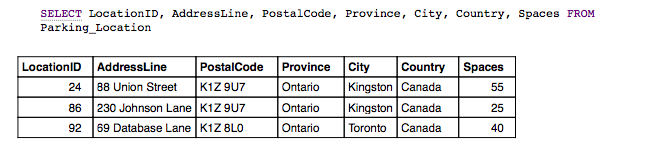
## Monthly Invoice Handle Form

The monthlyInvoiceHandle.php form is called when the user clicks the Submit button in the Monthly Invoice section of the admin homepage. In the admin homepage, the user specifies the member and start/end dates that they would like to retrieve the Invoice for. These values are passed to monthlyInvoiceHandle and the following query is executed.



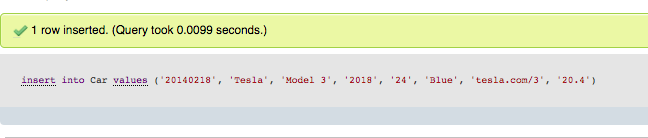
## Add Car Form

Within this form admins will add a new car to the fleet. The only SQL statement executed here is to dynamically populate the location to associate the new car with as seen below.

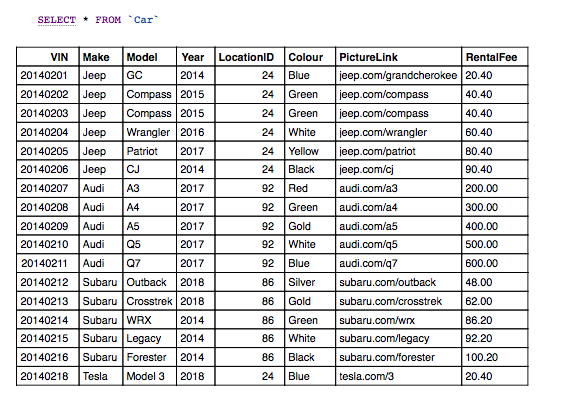


## Add Car Handle Form

The addCarHandle form contains all the admin entered information from the addCarForm file. The addCarHandle performs the following query to add the car to the fleet with the admin entered car information.



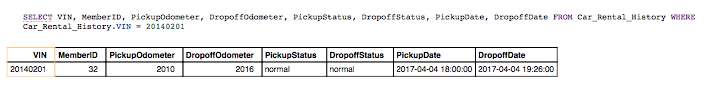
Doing a dump of the table gives the following results.



As can be seen above, the new car (Tesla Model 3) was added to the fleet.

## Car Rental History Handle Form

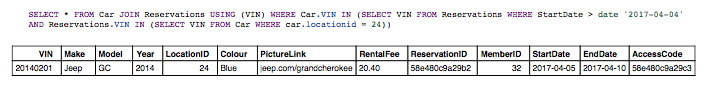
The carRentalHistoryHandle file is called from the ‘View Rental History for Cars’ section of the admin homepage. Within the admin homepage admins specify the car they would like to retrieve rental information about and the chosen car’s VIN is passed to the carRentalHistoryHandle form. The following query is then performed.



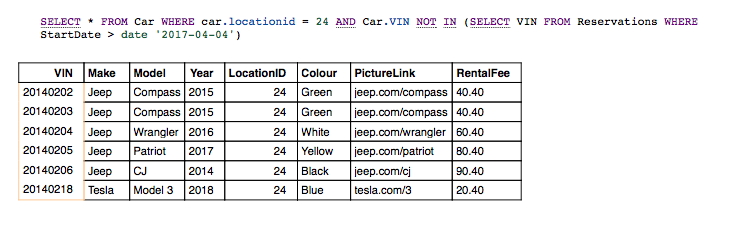
## Available Cars at Location Form

From the admin homepage admins select the parking location they would like to view information about the current available and reserved cars. The following queries are then performed.

For cars that have current reservations:

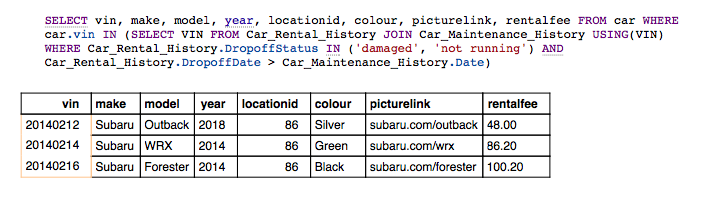


For cars that do not have current reservations:



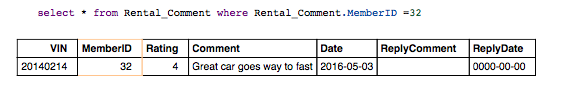
## Damaged or Need to be Repaired Cars Handle

Upon admins entering the following form the following SQL query will be executed.



## Rental Comment Handle Form

Upon users selecting a member and hitting submit in the ‘View Rental Comments’ section the following query will be executed.



Since the admin has not replied to the users comment a ‘Reply to Comment’ button will be seen. If the admin hits this button it will take them to the rentalCommentObj file which will ask them to enter their reply.

## Rental Comment Obj Form

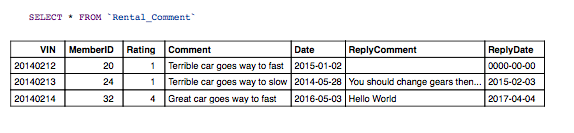
Upon entering this form, admins will see the original comment and will have a text field to reply to the comment. No SQL queries are executed here.

## Insert Reply Comment Form

Upon admins entering their response to the user and pressing submit the insertReplyComment form will update the record in the database using the following query.



The table dump can be seen below.

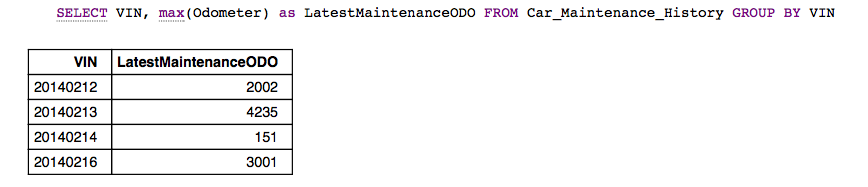


As can be seen above the replyComment and ReplyDate fields were updated for the specific user comment.

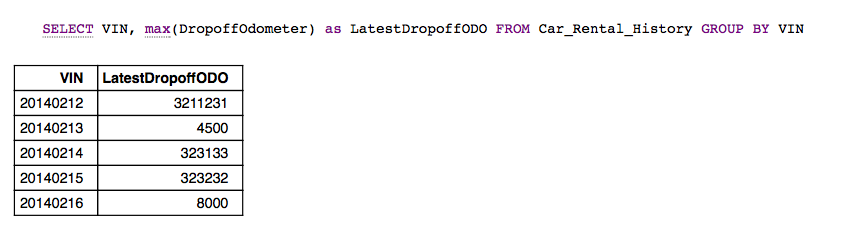
## Cars Distance Since Maintenance Handle

The carsDistanceSinceMaintenanceHandle form takes in the user input for the odometer value of the cars since maintenance from the admin homepage. With this value, it then performs the followings queries.

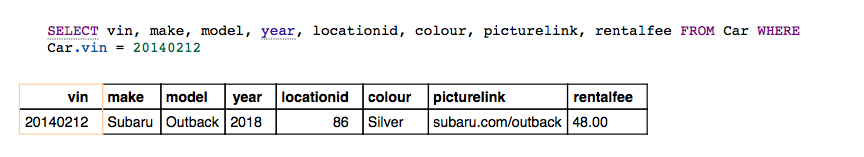
First it queries all unique VINS in Car\_Maintenance\_History and returns its highest odometer value (its latest one).



Next the following query is performed to query all unique VINS in Car\_Rental\_History and return its highest dropoff odometer value (its latest one).



Finally, the results from both queries are compared to one another through various php logic and finally the following query is executed to query all unique car information for the relevant VIN in the array. The query shown below just shows this for one VIN of 20140212.



To get the final variable seen on the form which is the km travelled since last maintenance the first two queries seen above were compared to one another and various php logic was used to find this value. This value will be displayed for the given car info seen above.

# Logout

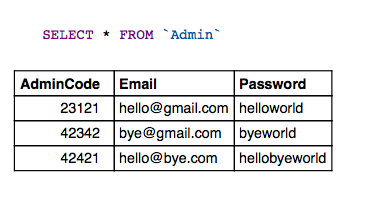
Logout does not perform any SQL commands. It solely destroys the session and redirects to the Login page.

# Appendix

## SQL Dumps of All Tables Preloaded with the KTCS\_DB script

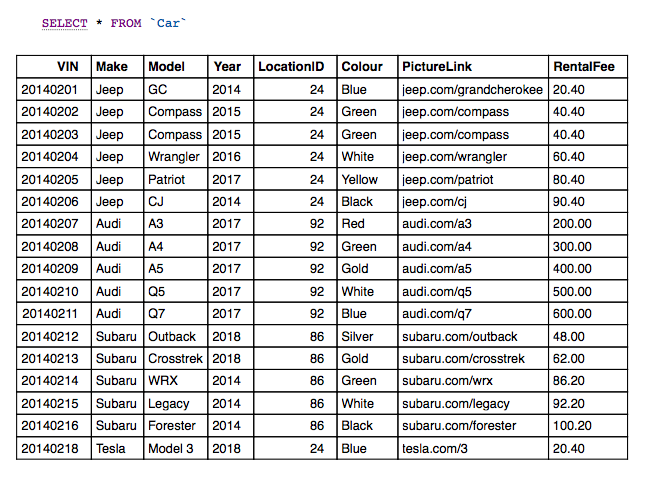
### Admin Table

The SQL table dump can be seen below.



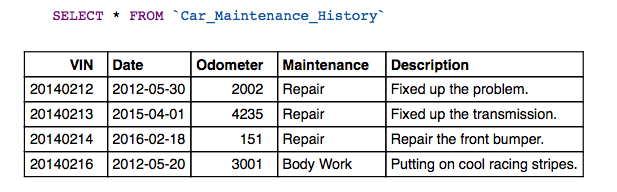
### Car Table

The SQL table dump can be seen below.



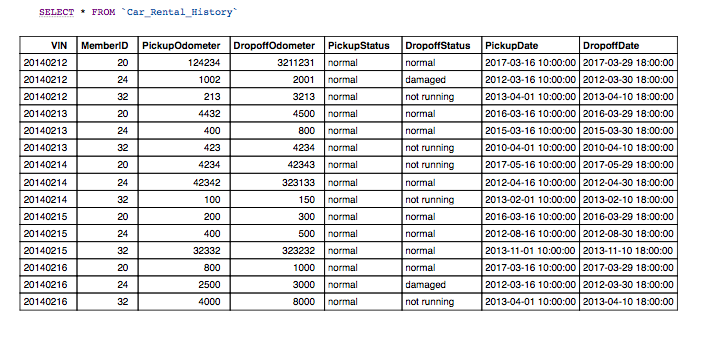
### Car\_Maintenance\_History Table

The SQL dump can be seen below



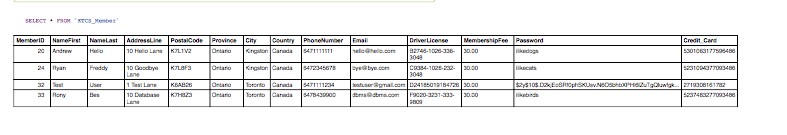
### Car\_Rental\_History Table

The SQL dump can be seen below.



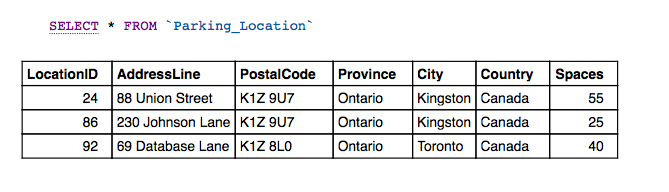
### KTCS\_Member Table

The SQL dump can be seen below.



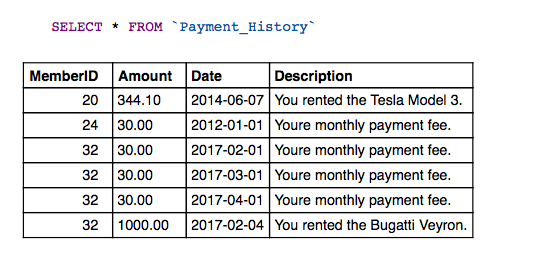
### Parking\_Location Table

The SQL dump can be seen below.



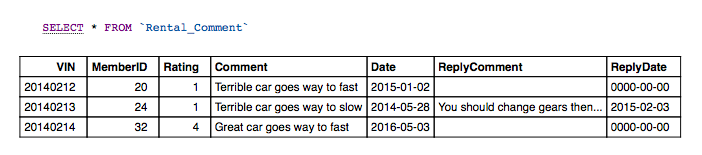
### Payment\_History Table

The SQL dump can be seen below.



### Rental\_Comment Table

The SQL dump can be seen below.



### Reservations Table

The SQL dump can be seen below.

