Team 23 Phase 2 Abstract Code w/SQL Report

Table of Contents:

Login
Search for vehicles
View vehicles detail
Add vehicle/purchase transaction
Perform sales
Look up customers
Add new customers
Add repairs/recalls and new vendors
Edit repairs/recalls status and view descriptions
View Seller History Report
View Inventory Age Report
View Average Time in Inventory Report
View Price Per Condition Report
View Repairs Statistics Report
View Monthly Sales Report

MySQL database schema is used.

Login

Abstract code

- User sees a ***Login*** button on the Public **Search** form and clicks on it.
- A login prompt asking for username('$username') and password('$password') appears.
- User enters username and password in the corresponding input fields.
- Data validation is done on username and password fields.
- If the data validation is successful:
    - When "***Enter***" button is clicked by the user:

```
SELECT passwords FROM Users WHERE username LIKE
'$username';
```

    - If '$username' belongs to a user in User table(query returns a user record):
        - if user.password!='$password':
            - redirect to **Login** form with error message "Incorrect credentials for the user!"
        - if user.password=='$password':
            - Go to **Search** form for logged-in users
            - Store login information as session variable '$sessionID'
    - else display on **Login** form"User does not exist!"
- Else an error message is displayed on the **Login** form that says "Username and password input fields are invalid!"

Search for Vehicle

Abstract Code:

- Display ***Search for vehicle, Login*** buttons in the public-facing search screen.
- Query Inventory vehicle and calculate the total number of vehicles available for purchase.
- Display " There are '$calculation_result' vehicles available for sale" prominently.

```
-- cal vehicles available for sale
SELECT COUNT(VIN) AS calculation_result FROM InventoryVehicle;
```

- When user clicks ***Login*** button - Run **Log in** subtask.

Table of Contents

- Once a user is logged in, instead of the ***Login*** button, ***Logout*** button appears in the search page for that user.
- When logged-in user clicks ***Logout*** button - Invalidate login session and go back to the **Login** form
- When User clicks ***Search for vehicle*** button, check user login status by *Username:*
    - If no login information detected, run **Public search** task:Display **Search for vehicle** form with drop-downs of *Vehicle type, Manufacturer, Model year, Color(s) and a "Search by keyword" text input.*
    - if user is logged-in user:
        - If user is sales people or inventory clerks - Run **Inventory clerks and salespeople search** task
            - Display **Search for vehicle** form with drop-downs of *Vehicle type, Manufacturer, Model year, Color(s),* a "*Search by keyword*" text input and a "*Search by VIN*" text input.
        - If user is Manager or Mr. Burdell - Run **Manager/Mr. Burdell search** task
            - Display **Search for vehicle** form with drop-downs of *Vehicle type, Manufacturer, Model year, Color(s), Sold/Unsold*, a "*Search by VIN*" text input and a "*Search by keyword*" text input.
            - Display ***View Seller History Report, View Inventory Age Report, View Average Time in Inventory Report, View Price Per Condition Report, Repair Statistics Report*** and ***View Monthly Sales Report*** buttons

```
-- populate dropdowns for search for vehicle
SELECT type_name FROM Vehicle Types;
SELECT manu_name FROM Manufacturer;
SELECT model_year FROM InventoryVehicle INNER JOIN Vehicle
ON Vehicle.VIN= InventoryVehicle.VIN;
SELECT color FROM Vehicle_Color;
```

- Inputted data on user selected search criteria, texted VIN or texted "Keyword" is used to retrieve search results as follows:
    - Lookup "Vehicles" table and retrieve results that match all the criteria and contain the keyword(AND logic for all selected criteria and keyword).

Table of Contents

- Query information about sales transaction and repair for the vehicles on the search result, by looking up on “Sales Transaction” and “Repair” tables.
- If the user is a public user or sales people, run Retrieve and display results for salespeople and public search subtask:

    - Lookup InventoryVehicle and Vehicle table and retrieve results that match all the criteria and contain the keyword(AND logic for all selected criteria and keyword).
    - Sort vehicle by VIN
    - Display VIN, Vehicle type, Model Year, Manufacturer, Model, Color(s), Mileage and Sales Price for vehicles on the lookup result.

```
-- salespeople and public user search
-- application provide $vehicle_type, $manufacturer, $vehicle_my, $vehicle_color and $KeyWord as search criteria

SELECT InventoryVehicle.VIN, Vehicle.type_name, Vehicle.model_year, Vehicle.manu_name, Vehicle.model_name, Vehicle_Color.color, Vehicle.mileage, InventoryVehicle.sales_price
FROM InventoryVehicle INNER JOIN Vehicle ON Vehicle.VIN= InventoryVehicle.VIN
INNER JOIN Vehicle_Color ON InventoryVehicle.VIN= Vehicle_Color.VIN
WHERE Vehicle.type_name="$vehicle_type" AND Vehicle.model_year ="$vehicle_my" AND Vehicle.manu_name ="$manufacturer" AND Vehicle_Color.color ="$vehicle_color" AND ( Vehicle.type_name = "$keyword" OR Vehicle.model_year ="$keyword" OR Vehicle.manu_name ="$keyword"OR Vehicle_Color.color ="$keyword" OR Vehicle.mileage ="$keyword" OR Vehicle.model_name = "$keyword" OR Vehicle.optional_desc="$keyword")
ORDER BY VIN;
```

- If the user is inventory clerk, run **Retrieve and display results for inventory clerks** subtask:
    - Lookup Vehicle, InventoryVehicle and Vehicle_Color table table and retrieve results that match all the criteria and contain the keyword(AND logic for all selected criteria and keyword).
    - Lookup “RepairedVehicle” table and retrieve results that match all the criteria and contain the keyword(AND logic for all selected criteria and keyword).
    - Sort vehicle by VIN
    - Display VIN, Vehicle type, Model Year, Manufacturer, Model, Color(s), Mileage and Sales Price for vehicles on the lookup result.

Table of Contents

```
-- Inventory clerk search
-- application provide $VIN, $vehicle_type, $manufacturer, $vehicle_my,
$vehicle_color and $KeyWord as search criteria for inventory clerk search

SELECT InventoryVehicle.VIN, Vehicle.type_name, Vehicle.model_year,
Vehicle.manu_name, Vehicle.model_name, Vehicle_Color.color, Vehicle.mileage,
InventoryVehicle.sales_price
FROM InventoryVehicle INNER JOIN Vehicle ON Vehicle.VIN= InventoryVehicle.VIN
INNER JOIN Vehicle_Color ON InventoryVehicle.VIN= Vehicle_Color.VIN
UNION
SELECT RepairVehicle.VIN, Vehicle.type_name, Vehicle.model_year,
Vehicle.manu_name, Vehicle.model_name, Vehicle_Color.color, Vehicle.mileage, ""
FROM RepairVehicle INNER JOIN Vehicle ON Vehicle.VIN= RepairVehicle.VIN
INNER JOIN Vehicle_Color ON Vehicle.VIN = Vehicle_Color.VIN
WHERE Vehicle.VIN="$VIN" AND Vehicle.type_name="$vehicle_type" AND Vehicle.
model_year ="$vehicle_my" AND Vehicle.manu_name ="$manufacurer" AND
Vehicle_Color.color ="$vehicle_color" AND (Vehicle.type_name = "$KeyWord" OR
Vehicle.model_year ="$KeyWord" OR Vehicle.manu_name ="$KeyWord" OR
Vehicle_Color.color ="$KeyWord" OR Vehicle.mileage ="$KeyWord" OR
Vehicle.model_name = "$KeyWord" OR Vehicle.Optional_Desc="$KeyWord")
ORDER BY VIN;
```

- If the user is Mr. Burdell, run **Retrieve and display results for Manager/Mr. Burdell** subtask:
  - Lookup Vehicle, InventoryVehicle and Vehicle_Color table and retrieve results that match all the criteria and contain the keyword(AND logic for all selected criteria and keyword).
  - Sort vehicle by VIN
  - Display VIN, Vehicle type, Model Year, Manufacturer, Model, Color(s), Mileage and Sales Price for vehicles on the lookup result.

```
-- Manager, Mr. Burdell search
-- application provide $VIN, $vehicle_type, $manufactures, $vehicle_my,
$vehicle_color -- as search criteria for Manager/Mr. Burdell search

SELECT InventoryVehicle.VIN, Vehicle.type_name, Vehicle.model_year,
Vehicle.manu_name, Vehicle.model_name, Vehicle_Color.color, Vehicle.mileage,
InventoryVehicle.sales_price
FROM Vehicle inner join vehicle_color on vehicle.vin=vehicle_color.vin left JOIN
InventoryVehicle ON Vehicle.VIN= InventoryVehicle.VIN
WHERE Vehicle.VIN="$VIN" AND Vehicle.type_name="$vehicle_type" AND Vehicle.
```

Table of Contents

```
model_year ="$vehicle_my" AND Vehicle.manu_name ="$manufactures" AND
Vehicle_color.color ="$vehicle_color" AND (Vehicle.type_name = "$KeyWord" OR
Vehicle.model_year ="$KeyWord" OR Vehicle.manu_name ="$KeyWord" OR
Vehicle_Color.color ="$KeyWord" OR Vehicle.mileage ="$KeyWord" OR
Vehicle.model_name = "$KeyWord" OR Vehicle.Optional_Desc="$KeyWord")
ORDER BY VIN;
```

- If there is no vehicle matching the search keywords:
    - Display: “Sorry, it looks like we don’t have that in stock!”

View Vehicle Detail

Abstract Code:

- User selected a vehicle from vehicle search results.
- Run **View Vehicle Detail** task:
    - Identify the user by login status and username.
    - If the user is a public user, run **Public user view** subtask:
        - Query Vehicle, InventoryVehicle and Vehicle_Color table for information about the selected vehicle
        - Display the query result with *VIN, vehicle type, Model Year, Manufacturer, color(s), mileage, sales price,* and *the description* of the car

```
-- public user view
-- application provide $VIN based on user’s selection
SELECT Vehicle.VIN, Vehicle.type_name, Vehicle.model_year, Vehicle.manu_name,
Vehicle.mileage, Vehicle.optional_desc, Vehicle_Color.color,
InventoryVehicle.sales_price
FROM Vehicle INNER JOIN Vehicle_Color ON Vehicle.VIN=Vehicle_Color.VIN
INNER JOIN InventoryVehicle ON Vehicle.VIN=InventoryVehicle.VIN
WHERE VIN=”$VIN”;
```

    - If the user is inventory clerk, run **Inventory Clerk view** subtask:
        - Query Vehicle, InventoryVehicle and Vehicle_Color table for information about the selected vehicle and display *VIN, vehicle type, Model Year, Manufacturer, color(s), mileage, sales price and the description of the car.*
        - Find and display KBB_Price(original purchase price) associated with the vehicle by querying Purchase_transaction table.

Table of Contents

- Query Repair table to find total repair cost for the selected vehicle (calculate and display total of all repair costs).
- Display each repair associated with the selected vehicle using ***view repairs/recalls*** subtask:
  - Display all repairs with *vendor, start date, end date, status, cost,* and *the recall number* if it is a recall.
  - To edit the repair status or to view description of repair, click on **Edit repairs/recalls status and view descriptions** button and it directs to **Edit repairs/recalls status and view descriptions** task.
    - Display an ***Add repair*** button:
  - If user presses the ***Add repair*** button, it directs to **Add repair/recall** task.

```
-- Inventory view vehicle detail
-- application provide $VIN based on user's selection

SELECT Vehicle.VIN, Vehicle.type_name, Vehicle.model_year, Vehicle.manu_name,
Vehicle.mileage, Vehicle.optional_desc, Vehicle_Color.color, InventoryVehicle.sales_price
FROM Vehicle INNER JOIN Vehicle_Color ON Vehicle.VIN=Vehicle_Color.VIN
INNER JOIN InventoryVehicle ON Vehicle.VIN=InventoryVehicle.VIN
WHERE VIN="$VIN";

SELECT KBB_Price FROM P_Transaction WHERE VIN="$VIN";
SELECT Total_Cost, vendor_name, start_date, end_date, repair_status,
NHTSA_Recall_Number
FROM Repairs INNER JOIN Determine ON Determine.repairID=Repairs.repairID
WHERE Determine.VIN="$VIN";
```

- If the user is a Salespeople, run **Salespeople view** subtask
  - Query Vehicle table for information about the selected vehicle by VIN and display the query result with VIN, vehicle type, Model Year, Manufacturer, color(s), mileage, sales price, and the description of the car
  - Display a "Sell" button
    - If user press the "Sell" button, direct to task Perform Sales.

```
-- salespeople view vehicle detail
-- application provide $VIN based on users selection
SELECT Vehicle.VIN, Vehicle.type_name, Vehicle.model_year, Vehicle.manu_name,
```

Table of Contents

```
Vehicle.mileage, Vehicle.optional_desc, Vehicle_Color.color,
InventoryVehicle.sales_price
FROM Vehicle INNER JOIN Vehicle_Color ON Vehicle.VIN=Vehicle_Color.VIN
INNER JOIN InventoryVehicle ON Vehicle.VIN=InventoryVehicle.VIN
WHERE VIN="$VIN";
```

- If the user is a Manager, run **Manager view** subtask
  - Query for information about the selected vehicle from the Vehicle table by VIN and display *VIN, vehicle type, Model Year, Manufacturer, color(s), mileage, sales price, the description of the car*
  - Query for information of total cost and repairs list from Repair table for the selected vehicle.
  - Display each repair associated with the selected vehicle using ***view repairs/recalls*** subtask:
    - Display all repair with *vendor, start date, end date, status, cost,* and *the recall number* if it is a recall
  - Calculate and display total of all repair costs.
  - Query the Inventory clerk table and Purchase Transaction table to find who purchased the vehicle.
  - Find and display the *First Name* and *Last Name* of the inventory clerk who purchased the vehicle.
  - Query the Customer table and Purchase transaction table for finding who sold the vehicle to Burdell's.
  - Find and display the seller's information:
    - Display *First Name, Last Name, Phone, Address* and *email* (if provided) for individual customer.
    - Display *B_Name, Primary Contact, phone, Address* and *email* (if provided) for business customers.
  - Query the Purchase transaction table and display Vehicle condition, KBB_Price and PurchaseDate for the selected vehicle.
  - If the selected vehicle has been sold from Burdell's,
    - Query for the information about Sales transaction and the Salespeople to find who sold the vehicle
    - Find and display Sales people's *First Name* and *Last Name from Salespeople table.*
    - Query information about Sales transaction and Customer to find which customer bought the vehicle from Burdell's.

Table of Contents

- Find and display the buyer's information
  - Display *First Name, Last Name, Phone, Address* and *email* (if provided) for individual customer
  - Display *B_Name, C_name, title, home, Address* and *email* (if provided) for business customer

```
-- manager view.application provide $VIN based on users selection
-- display vehicle related information
SELECT VIN, type_name, model_year, manu_name, mileage, optional_desc
FROM Vehicle WHERE VIN="$VIN";
SELECT color FROM Vehicle_Color WHERE VIN="$VIN";
SELECT KBB_Price FROM P_Transaction WHERE VIN="$VIN";
SELECT Sales_Price FROM InventoryVehicle WHERE VIN="$VIN";
SELECT Total_Cost, vendor_name, start_date, end_date, repair_status,
NHTSA_Recall_Number
FROM Repairs INNER JOIN Determine ON Determine.repairID=Repairs.repairID
WHERE Determine.VIN="$VIN";

-- display Inventory clerk's info and customer's info
SELECT Users.first_name, Users.last_name FROM P_Transaction INNER JOIN Users ON
P_Transaction. inventory_clerk_username= Users.username
WHERE P_Transaction.VIN="$VIN";
SELECT Customer.phone, Customer.street, Customer.city, Customer.state,
Customer.postal_code,  Customer.email FROM P_Transaction INNER JOIN Customer ON
P_Transaction. customerID= Customer.customerID
WHERE P_Transaction.VIN="$VIN";
-- display individual who sold the vehicle
SELECT Individual.first_name, Individual. last_name FROM P_Transaction INNER JOIN
Individual ON P_Transaction.customerID= Individual.customerID
WHERE P_Transaction.VIN="$VIN";
-- display business who sold the vehicle
SELECT Business.b_Name, Business.c_name, Business.title FROM P_Transaction INNER
JOIN Business ON P_Transaction. customerID= Business.customerID
WHERE P_Transaction.VIN="$VIN";

-- If vehicle has been sold
-- display sales people's info and customer's info
SELECT Users.first_name, Users.last_name
FROM SalesTransaction INNER JOIN Users
ON SalesTransaction.salespeople_username= Users.username
WHERE SalesTransaction.VIN="$VIN";
SELECT Customer.phone, Customer.street, Customer.city, Customer.state,
Customer.postal_code,  Customer.email FROM SalesTransaction INNER JOIN Customer ON
```

Table of Contents

```
SalesTransaction.customerID= Customer.customerID
WHERE SalesTransaction.VIN="$VIN";
-- display individual sold the vehicle
SELECT Individual.first_name, Individual.last_name FROM SalesTransaction INNER JOIN
Individual ON SalesTransaction.customerID= Individual.customerID
WHERE SalesTransaction.VIN="$VIN";
-- display business sold the vehicle
SELECT Business.b_Name, Business.c_name, Business.title FROM SalesTransaction INNER
JOIN Business ON SalesTransaction.customerID= Business.customerID
WHERE SalesTransaction.VIN="$VIN";
```

- If the user is Mr. Burdell, run **Mr. Burdell view** subtask:
  - Display "***View as Salespeople***", "***View as Inventory Clerks***" and ""***View as Manager***" buttons
  - If user press ***View as Salespeople*** button, run **Salespeople view** subtask
  - If user press ***View as Inventory Clerks*** button, run **Inventory clerk view** subtask
  - If user press ***View as Manager*** button, run **Manger view** subtask

---

Add vehicles/purchase transaction

Abstract code:

During purchase transactions:

- click on ***Add vehicle*** button.
- the **Add vehicle** form appears:
  - Enter the VIN ('$VIN')
  - Enter the type of vehicle from the list in database ('$vehicle_type')
  - Enter the manufacturer name from the list. ('$manu_name')
  - Enter the model name ('$model_name')
  - Enter the model year ('$model_year')
    - Check that model year <= current year +1
    - Check that model year includes century digits.
  - Enter the mileage (odometer reading). ('$mileage')
  - Enter the description (optional). ('$optional_desc')
  - Enter the colors from the list of generic color names. ('$color1', '$color2', '$color3',...)
- Click on the ***Save*** button to save the vehicle data into the database.

```
INSERT INTO Vehicle(VIN, type_name, manu_name, model_name,
model_year, mileage, optional_desc)
VALUES
```

Table of Contents

```
 ( '$VIN', '$vehicle_type', '$manu_name', '$model_name',  '$model_year',
'$mileage','$optional_desc' );
-- to add colors to database
INSERT INTO Vehicle_Color(VIN, color)
VALUES
( '$VIN', '$color1'), ( '$VIN', '$color2'), ( '$VIN', '$color3');
```

- To save the details of purchase transaction,
    - Enter the purchase date ('$purchase_date')
    - Enter the vehicle condition ('$vehicle_condition')
    - Enter the Blue book value ('$kbb_price')
    - Enter the username of the inventory clerk who handled the purchase transaction.('$inventory_clerk_username')
    - If the seller is a new customer, his details are added to the database using ***Add_new_customer*** button. If the seller is an already existing customer, there is no need to again add his details. Once the seller information is added (if he is a new customer), the customer ID is retrieved as '$customerID' .
    - Click on the ***Save*** button to add the data to database.

```
-- when customer is an individual with driver_license = '$driver_license'
SELECT customerID FROM Individual WHERE driver_license LIKE
'$driver_license';
-- when customer is a business with TIN = '$TIN'
SELECT customerID FROM Business WHERE TIN LIKE  '$TIN';
-- adding purchase data to DB
INSERT INTO P_transaction(purchase_date, VIN, customerID,
vehicle_condition, kbb_price, inventory_clerk_username)
VALUES
 ( '$purchase_date', '$VIN', '$customerID', '$vehicle_condition',
'$kbb_price', '$inventory_clerk_username');
```

Perform sales

Abstract Code:

- Sales people logged in and press ***Search for vehicle*** button – Run **Search for vehicle** task
- Sales people select the vehicle for sale – Run **View vehicle detail** task
- Sales people click ***Sell*** button – Load **Sales order** form with a ***Look up Customer*** button and a confirm date text input

Table of Contents

- Sales people press ***Look up Customer*** button on **Sales order** form – Run **Look up Customer** task
- If customer doesn't exist, run **Add customer** task
- Sales people selected the customer as a buyer

Sales people confirm the sale by entering sales date on the **Sales order** form – Update Inventory vehicle table and Sales transaction table

```
-- application provides $VIN, $customerID based on user's click and $username by --
reading user's login information
-- if the customer is an individual with driver's license = '$drivers_license', after adding
him to the database using add_customer task(if new customer):
-- Update Sales transaction table
INSERT INTO  SalesTransaction
(sales_date, VIN, customerID, salespeople_username)
VALUES
(CURDATE(), "$VIN", (SELECT customerID FROM Individual WHERE driver_license
LIKE  '$driver_license'), "$username");
-- if the customer is a business with TIN = '$TIN', after adding the Business details to
the database using add_customer task(if new customer):
-- Update Sales transaction table
INSERT INTO  SalesTransaction
(sales_date, VIN, customerID, salespeople_username)
VALUES(CURDATE(), "$VIN", (SELECT customerID FROM Business WHERE TIN
LIKE  '$TIN'), "$username");

-- Update Inventory vehicle table
DELETE FROM InventoryVehicle WHERE VIN="$VIN";
```

Look up customers

Abstract Code:

- User click ***Look up Customer*** button on **Sales Order** form (for sales people in vehicle sales transaction) or in **Add Vehicle** form (for inventory clerks in vehicle purchase transaction)
- Display **Look up Customer** form with *driver's license number/tax ID* text input and a ***Search*** button
- User input customer *driver's license number* or *tax ID* and press the ***Search*** button
- Query Customer table and lookup the customer whose *driver's license* or *tax ID* matches the text input.

Table of Contents

- Display lookup result: customer whose driver's license or tax ID matches the input, if such a customer exists.
- If no result is found, show ***Add new customer*** button and display: "Customer does not exist, please add new customer"
    - l If customer pressed ***Add new customer*** button, direct to **Add new customer** task

```
-- application get $DL_No or $TIN from user input
-- for $TIN input to lookup business customer
SELECT Business.TIN, Business.b_name, Business.c_name, Business.title,
Customer.phone, Customer.street, Customer.city, Customer.state,
Customer.postal_code
FROM Business INNER JOIN Customer ON Business.customerID=
Customer.customerID
WHERE Business.TIN="$TIN";
-- for $DL_No to lookup individual customer
SELECT Individual.driver_license, Individual.first_name, Individual.last_name,
Customer.phone, Customer.street, Customer.city, Customer.state,
Customer.postal_code
FROM Individual INNER JOIN Customer ON Individual.customerID=
Customer.customerID
WHERE Individual.driver_license="$DL_No";
```

Add new customers

Abstract code

- To add the new customer, enter the following details on **Add new customers** form:
    - Enter phone number('$phone').
    - Enter email address (optional) ('$email').
    - Enter street in address ('$street')
    - Enter city in address ('$city')
    - Enter state in address ('$state')
    - Enter postal code in address ('$postal code')
    - Click on the ***Save*** button to save the customer data into the database.

```
INSERT INTO Customer(phone, email, street, city, state, postal_code)
VALUES
( '$phone', '$email', '$street', '$city', '$state', '$postal code' );
```

- If the customer is an individual:

Table of Contents

- Obtain customerID from Customer table using LAST_INSERT_ID()
- Enter driver's license number ('$driver_license')
- Enter first name('$first_name')
- Enter last name ('$last_name')
- Click on the ***Save*** button to save the individual customer data into the database.

```
INSERT INTO Individual(driver_license, first_name, last_name, customerID)
VALUES
( '$driver_license','$first_name','$last_name', LAST_INSERT_ID() );
```

- If the customer is a business:
  - Enter Tax Identification Number ('$TIN')
  - Enter business name ('$b_name')
  - Enter primary contact name ('$c_name')
  - Enter primary contact title ('$title')
  - Click on the ***Save*** button to save the business customer data into the database.

```
INSERT INTO Business(TIN, b_name, c_name,title, customerID)
VALUES
( '$TIN', '$b_name', '$c_name', '$title', LAST_INSERT_ID() );
```

Add repairs/recalls and new vendors

Abstract code:

For each new repair determined for the vehicle with VIN='$VIN':

- Add the vehicle to RepairVehicle to determine the repairs needed.
- Add the determined repair to the database:
  - Enter the VIN of vehicle associated ('$VIN')
  - Enter the username of the inventory clerk who determined the repair ('$inventory_clerk_username')
  - Click on the ***Save*** button to insert the data into the database
  - Obtain the repairID for the determined repair ('$repairID') using LAST_INSERT_ID().

```
-- add vehicle VIN to RepairVehicle so as to facilitate the clerk to determine the repairs.
INSERT INTO RepairVehicle(VIN)
VALUE ( '$VIN');
-- add the determined repairs to Determine table
INSERT INTO Determine (VIN, inventory_clerk_username)
```

Table of Contents

```
VALUES
( '$VIN', '$inventory_clerk_username');
-- obtain the repairID for the last repair determined
SELECT LAST_INSERT_ID() as repairID;
```

- If the determined repair is a recall:
- Check if the recall is already in the database by looking up the NHTSA recall campaign number '$NHTSA'.
- If a row is returned, then the repair can be linked to this recall existing in the database using '$NHTSA'.
- If no rows returned, add a new recall to the database using the fields under the add_new_recall heading in the **Add repairs/recalls and new vendors** form :
  - Enter NHTSA recall campaign number ('$NHTSA')
  - Enter recall description ('$recall_description')
  - Enter associated manufacturer name ('$manu_name')
- Save the recall by clicking on the ***Save*** button.

```
INSERT INTO  Recall (NHTSA_recall_number, manu_name,
recall_description )
        SELECT '$NHTSA', '$manu_name', '$recall_description'
        FROM dual WHERE NOT EXISTS(
        SELECT NHTSA_recall_number FROM Recall WHERE
NHTSA_recall_number='$NHTSA'
);
```

- Get quotes for the determined repair from different vendors. For each considered vendor with vendor_name = '$vendor_name':
  - Check if the vendor is already in the database by looking up the vendor name '$vendor_name'.
  - If no rows are returned, the vendor is not in the database, then enter the following details in the fields under the add new vendor heading in **Add repairs/recalls and new vendors** form :
    - Enter the name of the vendor performing the repair/recall ('$vendor_name').
      - Check that the vendor name is unique.
    - Enter the Phone number of vendor ('$vendor_phone').
    - Enter the street in Address of vendor ('$street').
    - Enter the city in Address of vendor ('$city')
    - Enter the state in Address of vendor ('$state')
    - Enter the postal_code in Address of vendor ('$postal_code')
    - Save the vendor information by clicking on the ***Save*** button.

Table of Contents

```
INSERT INTO Vendor(vendor_name, vendor_phone, street, city, state,
postal_code)
SELECT '$vendor_name', '$vendor_phone', '$street', '$city', '$state',
'$postal_code' FROM dual WHERE NOT EXISTS(
SELECT vendor_name FROM Vendor WHERE vendor_name like
'$vendor_name');
```

- Now the quote obtained by the inventory clerk with username as '$inventory_clerk_username' for the determined repair ('$repairID') from the vendor with vendor_name ='$vendor_name' is added to the database:

```
INSERT INTO Quotes(vendor_name, repairID,
inventory_clerk_username)
VALUES
( '$vendor_name', '$repairID', '$inventory_clerk_username' );
```

- Now that the quotes from different vendors are obtained for the determined repair and the corresponding recall number is also obtained (if applicable), the following details can be entered to the **Add repairs/recalls and new vendors** form :
  - Enter start date ('$start_date') and end date ('$end_date') of the repair.
    - Check that the start date is not after the end date.
  - Enter the status of the repair (‘pending’ by default for all new repairs).
  - Enter the total cost of the repair ('$total_cost') based on the quote chosen.
  - Enter description of the repair (‘$repair_description’).
  - Enter the vendor name ('$vendor_name') based on the quote chosen.
  - If repair is associated with a recall, the corresponding NHTSA Recall Campaign number is obtained as '$NHTSA'.
  - '$repairID' is obtained in the “determine repair” step.

```
INSERT INTO Repairs (repairID, start_date, end_date, repair_status,
total_cost, repair_description, vendor_name, NHTSA_recall_number)
VALUES
( '$repairID', '$start_date', '$end_date', 'pending',  '$total_cost',
'$repair_description’, ‘$vendor_name', '$NHTSA');
```

When a manufacturer’s recall happens:

- Add the recall to the system using the fields under the add_new_recall heading in the **Add repairs/recalls and new vendors** form:
    - Enter NHTSA recall campaign number ('$NHTSA')
    - Enter recall description ('$recall_description')
    - Enter associated manufacturer name ('$manu_name')
  - Save the recall by clicking on the ***Save*** button.

```
INSERT INTO  Recall (NHTSA_recall_number, manu_name,
```

Table of Contents

```
recall_description )
VALUES
( '$NHTSA', '$manu_name', '$recall_description' );
```

Edit repairs/recalls status and View descriptions.

Abstract code:

When an inventory clerk or manager or Mr.Burdell accesses the vehicle detail page and clicks on ***edit repairs/recalls status*** button:

- All repairs/recalls with status not yet “completed” (as “pending” or “in progress” status) for that vehicle with VIN = '$VIN' appears as a list.

```
SELECT vendor_name, start_date, end_date, repair_status, total_cost,
NHTSA_recall_number FROM Repairs
WHERE repair_status NOT LIKE 'completed'
AND
repairID IN (SELECT repairID FROM Determine WHERE VIN ='$VIN');
```

- The corresponding description for each repair with repairID ‘$repairID’ is populated in a popup when that repair is chosen from the list. If the repair is a recall, then the recall description is also populated in the same screen.

```
SELECT Repairs.repair_description, Recall.recall_description FROM Repairs
LEFT OUTER JOIN Recall ON Repairs.NHTSA_recall_number =
Recall.NHTSA_recall_number WHERE Repairs.repairID =  '$repairID' ;
```

- The status of the repair in the list with start date as '$start_date' and current status as ‘pending’ can be changed to ‘in-progress’, and the repair with current status as ‘in progress’ can be changed to ‘completed’ by the clerk by clicking on the ***Update status*** button on the repair. Repairs with current status “completed” can no longer be updated.

```
UPDATE Repairs SET repair_status = IF( repair_status = 'pending' ,'in
progress' ,'completed') WHERE repairID = '$repairID'  and repair_status IN ( 'in
progress','pending');
```

- Once the status of a repair/recall is changed to “completed”, that repair/recall is no longer accessible from ***Edit repairs/recalls status*** button. This is taken care of by the select query displaying the repairs/recalls for a vehicle.
- Once all repairs/recalls of a vehicle is in the status “completed“, the vehicle with VIN = ‘$VIN’ is added to inventory, and made available for public search and sale by the inventory clerk whose username is '$inventory_clerk_username'.

Table of Contents

- The sales price ('$sales_price') is calculated as 125% of the original purchase price (the price Burdell's paid to buy the car) combined with 110% of any repair costs also associated with the vehicle.

```
-- to add vehicle data into Inventory Vehicle
INSERT INTO InventoryVehicle(VIN, sales_price, inventory_clerk_username)
SELECT  '$VIN',
(SELECT ((1.25 * P_transaction.kbb_price)+ (1.1* (SELECT
SUM(Repairs.total_cost) AS total_repair_cost FROM Repairs LEFT OUTER
JOIN Determine ON Repairs.repairID = Determine.repairID WHERE
Determine.VIN = '$VIN' ))) AS sales_price FROM P_transaction INNER JOIN
Determine ON P_transaction.VIN = Determine.VIN),
'$inventory_clerk_username'
FROM dual WHERE NOT EXISTS
(SELECT Repairs.repairID FROM Repairs LEFT OUTER JOIN Determine ON
Repairs.repairID = Determine.repairID WHERE Determine.VIN = '$VIN' AND
Repairs.repair_status IN ('in progress', 'pending') ) ;
-- to add vehicle to NoRepairVehicle so as to make it easy to pull/make reports
from DB.
INSERT INTO NoRepairVehicle(VIN, repair_vehicle_VIN,
inventory_clerk_username)
VALUES ( '$VIN',  '$VIN', '$inventory_clerk_username');
```

View Seller History Report

Abstract code

- Logged-in User Manager or Mr.Burdell click ***View Seller History Report*** button that was displayed on the search page.
- Query for all Seller and the total number of vehicles they have sold to Burdell's from Purchase_transactions table and Customers table, the average price for the vehicles from Purchase_transactions table and the average number of repairs per vehicle from Repair table.

```
SELECT  b_name, COUNT(*) AS number_of_vehicle, AVG(kbb_price) AS
average_purchase_price, COUNT(repairID) / COUNT(*) AS
average_number_repairs_per_vehicle
FROM  P_Transaction AS P, Determine AS D, Customer AS C, Business AS B
WHERE P.VIN = D.VIN AND P.customerID =C.customerID AND P.customerID =
B.customerID
GROUP BY  b_name
ORDER BY COUNT(P.VIN) DESC, AVG(kbb_price) ASC;
```

Table of Contents

```
SELECT first_name, last_name, COUNT(*) AS number_of_vehicle, AVG(kbb_price)
AS average_purchase_price, COUNT(repairID) / COUNT(*) AS
average_number_repairs_per_vehicle
FROM   P_Transaction AS P, Determine AS D, Customer AS C, Individual AS I
WHERE P.VIN =D.VIN AND P.customerID =C.customerID AND P.customerID =
I.customerID
GROUP BY  first_name, last_name
ORDER BY COUNT(P.VIN) DESC, AVG(kbb_price) ASC;
```

- Open a new window (**View Seller History Report** form) to show the Query result. Each row shows one seller's name, the total number of vehicles this seller have sold to Burdell's, the average purchase price, the average number of repairs per vehicle.
- Sort above query results by the total number of vehicles sold descending and then by average purchase price ascending.

- If the average number of repairs per vehicle >= 5 for a seller, set this seller's row background to red.

```
HAVING  COUNT(average_number_repairs_per_vehicle) >= 5;
```

- If user clicks the ***Close*** button, close the window.

View Inventory Age Report

Abstract code

- Logged in User Manager or Mr.Burdell clicks ***View Inventory Age Report*** button that was displayed on the search page.
- Query for unsold Inventory vehicles from Inventory vehicle table and Sales Transaction table.
- Compute the Inventory age in days for every vehicle by the formula: Inventory age(in days) = current date – Purchase Date

Table of Contents

- Find the minimum, average, and maximum age of unsold Inventory vehicles, in days.

```
SELECT type_name, MIN(DATEDIFF(CURDATE(),purchase_date)) AS
min_inventory_age, MAX(DATEDIFF(CURDATE(),purchase_date)) AS
max_inventory_age, AVG(DATEDIFF(CURDATE(),purchase_date)) AS
average_inventory_age
FROM Vehicle AS V, InventoryVehicle AS I, P_Transaction AS P
WHERE V.VIN = I.VIN AND I.VIN  = P.VIN
GROUP BY  type_name;
```

- Open a new window to show the Query result. Each row shows one type of vehicle, the minimum, average, and maximum age of unsold vehicles in inventory, in days.
- If one vehicle type has no unsold vehicles, display “N/A” for minimum, average, and maximum age.
- If user clicks the ***Close*** button, close the window.

View Average Time in Inventory Report

Abstract code

- Logged in User Manager or Mr.Burdell click ***View Average Time in Inventory Report*** button that was displayed on the search page.
- Query for sold vehicles from Inventory vehicles table and Sales Transaction table.
- Compute the Average Time in Inventory in days by the formula: Average Time in Inventory (in days) = total (Sales date – Purchase Date) / total number of sold vehicles.

```
SELECT type_name, SUM(DATEDIFF(sales_date , purchase_date)) / COUNT(*) AS
average_time_in_inventory
FROM Vehicle AS V, P_Transaction AS P, SalesTransaction AS S
WHERE V.VIN = P.VIN AND S.VIN  = P.VIN
GROUP BY  type_name;
```

Table of Contents

```
```

- Open a new window to show the result. Each row shows one type of vehicle, the Average Time in Inventory for sold vehicles, in days.
- If one vehicle type has no sales history, display “N/A” for Average Time in Inventory.
- If user clicks the ***Close*** button, close the window.

View Price Per Condition Report

Abstract code

- Logged in User Manager or Mr.Burdell click ***View Price Per Condition Report*** button that was displayed on the search page.
- Query for Purchase Price (Purchase Price = kbbPrice) for each vehicle from Purchase Transaction table and Vehicle table.
- Compute the price per condition by the formula: Price Per Condition = total Purchase Price for vehicles with the Condition / total number of vehicles with the condition.

```
SELECT type_name, vehicle_condition, SUM(kbb_price)/COUNT(*) AS
price_per_condition
FROM Vehicle AS V, P_Transaction AS P
WHERE V.VIN=P.VIN
GROUP BY type_name, vehicle_condition;
```

- Open a new window to show the result. Each row shows one type of vehicle, Price Per Condition for each condition.
- If one vehicle type or/and condition have no purchase history, display “$0” for Price Per Condition.
- If user clicks the ***Close*** button, close the window..

View Repair Statistics Report

Table of Contents

Abstract code

- Logged in User Manager or Mr.Burdell click ***View Repair Statistics Report*** button that was displayed on the search page.
- Query for Vendor name, repair, total cost, start date, end date for each vendor from Repair Vehicle table and Repair table.
- Compute the average number of repairs per vehicle for a specific vendor by the formula: average number of repairs per vehicle for the vendor = total number of repairs for the vendor / total number of Repair Vehicles for the vendor.
- Compute the total length of time (in days) for a vendor by the formula: length of time (in days) = end date – start date. Sum all repair's length of time (in days) to get total length of time (in days) for the vendor.
- Compute the average length of time (in days) for a specific vendor by the formula: average length of time (in days) for the vendor = total length of time (in days) for the vendor / total number of repairs for the vendor.

```
SELECT vendor_name, COUNT(*) AS number_of_repairs,
sum(total_cost) AS grand_cost, (count(*)/(SELECT COUNT(DISTINCT VIN)
FROM Determine)) AS number_of_repairs_per_vehicle,
(AVG(DATEDIFF(end_date , start_date))) AS average_length_of_time
FROM Repairs
GROUP BY vendor_name;
```

- Open a new window to show the result. Each row shows one vendor, the number of repairs completed by the vendor, the total cost spent on completed repairs, the average number of repairs per vehicle completed by that vendor, and the average length of time (in days) to complete repairs by that vendor.
- If user clicks the ***Close*** button, close the window.

View Monthly Sales Report

Abstract Code

- User logged in as a Manager or Mr. Burdell and selected Monthly Sales Report
- Query for information about all vehicle "Sales transaction"
- Sort "Sales transaction" by SalesDate with year and month descending.
- Display a list for all "Sale transaction" by year and month descending.

Table of Contents

- Calculate and display the total number of vehicles sold by year and month descending.
- Calculate and display the total sales income by year and month descending.
- Query for information about each “Purchase transaction” and “Repair” corresponding to each “Sales transaction” from Purchase transaction table and Repair table.
- Calculate the total purchase price corresponding to the “Sales transactions” from Purchase transaction table.
- Calculate the total repair cost corresponding to the “Sales transactions” from Repair table.
- Calculate and display the total net income by subtracting total purchase price and repair cost from total sales income.
- Query for information about each “Salespeople” corresponding to each “Sales transaction” from SalesPeople table and Sales transaction table.
- Calculate the number of vehicles sold and the total sales by each “SalesPeople” by year and month.
- Sort the “SalesPeople” based on the total vehicle and then by total sales.
- Display “SalesPeople” first name and last name, the number of vehicles they sold and the total sales for the year and month.
- Determine and display the top sales person by the highest total sales by year and month.

```
-- list for all sales transactions:
SELECT sales_date, VIN FROM SalesTransaction
ORDER BY sales_date DESC;
-- cal and display total number of vehicle sold
SELECT  YEAR(sales_date) AS YEAR, MONTH(sales_date) as month, COUNT(VIN)
AS Total_Sold FROM SalesTransaction
GROUP BY YEAR(sales_date), MONTH(sales_date);
-- cal total sales income
SELECT SUM(InventoryVehicle.sales_price) AS
Total_Sales_Income,YEAR(sales_date) as year, MONTH(sales_date) as month
FROM InventoryVehicle
INNER JOIN SalesTransaction ON InventoryVehicle.VIN
= SalesTransaction.VIN
GROUP BY YEAR(sales_date), MONTH(sales_date);
-- cal total net income
```

Table of Contents

```
SELECT YEAR(SalesTransaction.sales_date) as year,
MONTH(SalesTransaction.sales_date) as month,(SUM(InventoryVehicle.sales_price)-
SUM(P_Transaction.kbb_price)-SUM(Repairs.Total_Cost)) AS Total_Net_Income
FROM SalesTransaction INNER JOIN InventoryVehicle ON InventoryVehicle.VIN =
SalesTransaction.VIN  INNER JOIN P_Transaction ON P_Transaction.VIN=
SalesTransaction.VIN INNER JOIN Determine ON
Determine.VIN=SalesTransaction.VIN INNER JOIN Repairs ON Repairs.repairID=
Determine.repairID
 GROUP BY YEAR(SalesTransaction.sales_date),
MONTH(SalesTransaction.sales_date);
-- drilldown report for top performing salespeople
-- application get the $selectmonth from user click
SELECT first_name, last_name, COUNT(SalesTransaction.VIN) AS NumofVeh,
SUM(sales_price) AS TotalSale FROM
SalesTransaction INNER JOIN InventoryVehicle
ON SalesTransaction.VIN= InventoryVehicle.VIN INNER JOIN
Users ON Users.username= SalesTransaction.salespeople_username
WHERE YEAR(SalesTransaction.sales_date) = YEAR("$selectmonth") AND
MONTH(SalesTransaction.sales_date) = MONTH("$selectmonth")
ORDER BY NumofVeh DESC, TotalSale DESC
LIMIT 1;
```

Table of Contents