Table of Contents

Login

Abstract Code:

- User enters *username*, *password* input fields.
- If form validation is successful for both *username* and *password* input fields, then:
    - When ***Login*** button is clicked:

```
USE DATABASE cs6400_sm19_team013;

SELECT
  login_password,
  role
FROM user
WHERE
  login_username='{username}';
```

        - If User record is found but user.login_password != *password* OR user not found:
            - Display error message.
        - Else:
            - More options appear on Search Page
                - All authenticated Users see Search by VIN option
                - If user.role == 'Inventory Clerk' OR 'Owner':
                    - Add Vehicle option appears
                    - Add Repair option appears
                - If user.role == 'Manager' OR 'Owner':
                    - ***Reports*** link appears on page

Search

Abstract Code:

- Public Facing Search Screen defined as:
    - Show **Login** form
    - Show number of cars available for purchase in the system

```
SELECT
  count(vehicle.vin) as vehicles_available
FROM vehicle
LEFT JOIN
(
  SELECT
    distinct(vin)  as vin
  FROM repair
  WHERE repair_status IN ('In Progress','Pending')
) vehicle_in_repair
ON vehicle.vin=vehicle_in_repair.vin
LEFT JOIN sale
  ON vehicle.vin=sale.vin
WHERE
  vehicle_in_repair.vin IS NULL AND
  sale.sales_date IS NULL;
```

- Prepopulate dropdown lists from database:
  - VehicleType

```
SELECT vehicle_type
FROM vehicle_type;
```

  - Manufacturer

```
SELECT manufacturer_name
FROM manufacturer;
```

- Static Lists predefined on customer specifications document
  - Pull Colors from list of colors predefined on customer specifications document (these will be defined on the UI itself)
  - Pull ModelYear from range 1900 to current year + 1 (these will be generated on the UI itself)
- User selects from none, one, or more of the following dropdowns:
  - VehicleType
  - Manufacturer
  - ModelYear
  - Color
- User additionally can enter a keyword in freeform field searching manufacturer_name, model_year, model_name, and description as entire text or substring

- If user inputs text in field, and selects from dropdown(s) and presses the ***Enter*** button
  - Find all vehicles based on user input and display

```
-- Query for search functionality
-- By default, form variables will be wildcard
SELECT
  vehicle.vin,
  vehicle.vehicle_type,
  vehicle.model_year,
  vehicle.manufacturer_name,
  vehicle.model_name,
  vehicle_color_grouped.color,
  vehicle.mileage,
  vehicle.sales_price,
  vehicle.model_name,
  vehicle.vehicle_description
FROM vehicle
LEFT JOIN (
  SELECT
    vin,
    GROUP_CONCAT(color ORDER BY color ASC SEPARATOR ',') as color
  FROM vehicle_color
  GROUP BY vin
) as vehicle_color_grouped
ON vehicle_color_grouped.vin=vehicle.vin
LEFT JOIN
(
  SELECT
    distinct(vin) as vin
  FROM repair
  WHERE repair_status IN ('In Progress','Pending')
) vehicle_in_repair
ON vehicle.vin=vehicle_in_repair.vin
LEFT JOIN sale
  ON vehicle.vin=sale.vin
WHERE
  vehicle_in_repair.vin IS NULL AND
  sale.sales_date IS NULL AND
  vehicle_type='{vehicle_type}' AND
  manufacturer_name='{manufacturer_name}' AND
  model_year='{model_year}' AND
  vehicle_type='{vehicle_type}' AND
  color LIKE '%{color}%' AND
  (manufacturer_name LIKE '%{keyword}%' OR
```

```
model_year LIKE '%{keyword}%' OR
vehicle_description LIKE '%{keyword}%' OR
model_name LIKE '%{keyword}%');
```

        - Selecting a result will load that vehicle's detail page
    - If no cars found, display:
        - 'Sorry, looks like we don't have that in stock!'
    - If no criteria selected and User presses the ***Enter*** button
        - Find all vehicles without pending repairs and display

```
-- Query if no criteria is entered
SELECT
  vehicle.vin,
  vehicle.vehicle_type,
  vehicle.model_year,
  vehicle.manufacturer_name,
  vehicle.model_name,
  vehicle_color_grouped.color,
  vehicle.mileage,
  vehicle.sales_price,
  vehicle.model_name,
  vehicle.vehicle_description
FROM vehicle
LEFT JOIN (
  SELECT
    vin,
    GROUP_CONCAT(color ORDER BY color ASC SEPARATOR ',') as color
  FROM vehicle_color
  GROUP BY vin
) as vehicle_color_grouped
ON vehicle_color_grouped.vin=vehicle.vin
LEFT JOIN
(
  SELECT
    distinct(vin)  as vin
  FROM repair
  WHERE repair_status IN ('In Progress','Pending')
) vehicle_in_repair
ON vehicle.vin=vehicle_in_repair.vin
LEFT JOIN sale
  ON vehicle.vin=sale.vin
WHERE
  vehicle_in_repair.vin IS NULL AND
  sale.sales_date IS NULL;
```

- If authenticated User:
    - Show Public Facing Search Screen
    - Provide additional freeform text field for searching by vin
        - User inputs vin in input field and clicks the ***Enter*** button
            - Find vehicle where vehicle.vin == vin and display

```
-- Searching for a specific VIN
SELECT
  vehicle.vin,
  vehicle_type,
  model_year,
  manufacturer_name,
  model_name,
  vehicle_color_grouped.color,
  mileage,
  sales_price
FROM vehicle
LEFT JOIN (
  SELECT
    vin,
    GROUP_CONCAT(color ORDER BY color ASC SEPARATOR ',') as
color
  FROM vehicle_color
  GROUP BY vin
) as vehicle_color_grouped
ON vehicle_color_grouped.vin=vehicle.vin
WHERE
  vehicle.vin='{vin}';
```

    - If authenticated user.role == 'Inventory Clerk':
        - Search results also include vehicles repairs 'pending' or 'in progress'

```
SELECT
  vehicle.vin,
  vehicle.vehicle_type,
  vehicle.model_year,
  vehicle.manufacturer_name,
  vehicle.model_name,
  vehicle_color_grouped.color,
  vehicle.mileage,
  vehicle.sales_price,
```

```
  vehicle.model_name,
  vehicle.vehicle_description
FROM vehicle
LEFT JOIN (
  SELECT
    vin,
    GROUP_CONCAT(color ORDER BY color ASC SEPARATOR ',') as
color
  FROM vehicle_color
  GROUP BY vin
) as vehicle_color_grouped
ON vehicle_color_grouped.vin=vehicle.vin
LEFT JOIN sale
  ON vehicle.vin=sale.vin
WHERE
  sale.sales_date IS NULL;
```

        - Show ***Add Vehicle*** button
    - Else if authenticated user.role == 'Salespeople':
        - Search results display same as public facing search

```
SELECT
  vehicle.vin,
  vehicle.vehicle_type,
  vehicle.model_year,
  vehicle.manufacturer_name,
  vehicle.model_name,
  vehicle_color_grouped.color,
  vehicle.mileage,
  vehicle.sales_price,
  vehicle.model_name,
  vehicle.vehicle_description
FROM vehicle
LEFT JOIN (
  SELECT
    vin,
    GROUP_CONCAT(color ORDER BY color ASC SEPARATOR ',') as
color
  FROM vehicle_color
  GROUP BY vin
) as vehicle_color_grouped
ON vehicle_color_grouped.vin=vehicle.vin
LEFT JOIN
(
```

```
  SELECT
    distinct(vin)  as vin
  FROM repair
  WHERE repair_status IN ('In Progress','Pending')
) vehicle_in_repair
ON vehicle.vin=vehicle_in_repair.vin
LEFT JOIN sale
  ON vehicle.vin=sale.vin
WHERE
  vehicle_in_repair.vin IS NULL AND
  sale.sales_date IS NULL;
```

- Else if authenticated user.role == 'Managers':
  - Show number of cars with repairs 'pending' or 'in progress'
  - Search results display same as public facing search plus: seller's contact information (except for driver license or tax id number), first and last name of inventory clerk that purchased the car, original purchase price, purchase date, total cost of repairs, repairs view of inventory clerk

```
-- vehicles NOT in repair (Available for sale)
SELECT
  count(vehicle.vin) AS number_of_vehicles
FROM vehicle
LEFT JOIN
(
  SELECT
    distinct(vin)  as vin
  FROM repair
  WHERE repair_status IN ('In Progress','Pending')
) vehicle_in_repair
ON vehicle.vin=vehicle_in_repair.vin
LEFT JOIN sale
ON sale.vin=vehicle.vin
WHERE
  vehicle_in_repair.vin IS NULL AND
  sale.sales_date is NULL;

-- vehicles in repair
SELECT
  count(vehicle.vin) AS number_of_vehicles
FROM vehicle
LEFT JOIN
(
```

```
  SELECT
    distinct(vin)  as vin
  FROM repair
  WHERE repair_status IN ('In Progress','Pending')
) vehicle_in_repair
ON vehicle.vin=vehicle_in_repair.vin
WHERE
  vehicle_in_repair.vin IS NOT NULL;
```

        - Provide additional dropdown for filtering search results by sold vehicles, unsold vehicles, or all vehicles
        - Search results also include vehicle repairs 'pending' or 'in progress'
        - Show ***Reports*** link
    - Else if authenticated user.role == 'Owner':
        - Search results also include vehicle repairs 'pending' or 'in progress'
        - Show number of cars with repairs 'pending' or 'in progress'

```
--to display count of vehicles in repair with status pending or
in progress
SELECT
  count(vehicle.vin) AS number_of_vehicles
FROM vehicle
LEFT JOIN
(
  SELECT
    distinct(vin)  as vin
  FROM repair
  WHERE repair_status IN ('In Progress','Pending')
) vehicle_in_repair
ON vehicle.vin=vehicle_in_repair.vin
WHERE
  vehicle_in_repair.vin IS NOT NULL;
```

        - Provide additional dropdown for filtering search results by sold vehicles, unsold vehicles, or all vehicles
        - Show ***Add Vehicle*** button
        - Show ***Reports*** link
- Else if not authenticated User:
    - Show Public Facing Search Screen

Search/Add Customer

Abstract Code:

- Provide a drop-down to select the type of customer with the following options
    - Individual
    - Business
- If 'Individual' is selected
    - Provide a text field to the user to input a *driver's license number*
    - User inputs a String in the input field and presses the ***Enter*** button
        - Display the customer for which individual.drivers_license == *user_input*. The following fields are displayed: Last name, first name, driver's license number, phone number, address

```
SELECT
  individual.individual_last_name,
  individual.individual_first_name,
  individual.driver_license_number,
  customer.phone_number,
  customer.street,
  customer.city,
  customer.state,
  customer.postal_code
FROM
  individual
INNER JOIN customer
ON individual.customer_id=customer.customer_id
WHERE
  individual.driver_license_number='{user_input}';
```

- If 'Business' is selected
    - Provide a text field to the user to input a *business tax identification number*
    - User inputs a String in the input field and presses the ***Enter*** button
        - Display the customer for which customer.business_tin == *user_input*. The following fields are displayed: Business name, business tax identification number, primary contact, title, phone number and address

```
SELECT
  business.business_name,
  business.tax_id_number,
  business.pc_title,
  business.pc_name,
  customer.phone_number,
```

```
  customer.street,
  customer.city,
  customer.state,
  customer.postal_code
FROM
  business
INNER JOIN customer
ON business.customer_id=customer.customer_id
WHERE
  business.tax_id_number='{user_input}';
```

- No writes to the Customer table happens if user selects the customer
- Add customer fields for adding a new customer are displayed when the Sales Order form or Purchase Vehicle form are selected
    - Show a drop-down to select the type of customer with the following two options
        - Individual
        - Business
    - If 'Individual' is selected from the drop-down, provide the following fields
        - First name
        - Last name
        - Driver's license
        - Phone number
        - Street
        - City
        - State
        - Postal code
        - Email

```
INSERT INTO customer
  VALUES(NULL, '{phone_number}', '{email}', '{street}',
'{city}', '{state}', '{postal_code}');

INSERT INTO individual
  VALUES('{driver_license_number}', LAST_INSERT_ID(),
'{individual_first_name}', '{individual_last_name}');
```

    - If 'Business' is selected from the drop-down, provide the following fields
        - Business name
        - Business tax identification number
        - Primary contact name
        - Primary contact title
        - Phone number
        - Street
        - City

- State
- Postal code

```
INSERT INTO customer
  VALUES(NULL, '{phone_number}', '{email}', '{street}',
'{city}', '{state}', '{postal_code}');

INSERT INTO business
  VALUES('{tax_id_number}', LAST_INSERT_ID(),
'{business_name}', '{pc_name}', '{pc_title}' );
```

Add Vehicle

Abstract Code:

- First populate the customer details following the actions in Search/add customer task
- To add a vehicle, provide the following fields to the user for input
  - VIN
  - Manufacturer (User selects one value from a dropdown populated using list of manufacturers in database)
  - Vehicle type (User selects one value from a dropdown populated using list of vehicle types in database)
  - Model year (User selects one value from a dropdown ranging from 1900 to (current_year + 1))
  - Color (User selects one or more values from a static dropdown)
  - Model name
  - Vehicle condition (User selects one value from dropdown populated using the following values: Excellent, Very Good, Good, Fair)
  - KBB value
  - Mileage
  - Description
  - Auto-populate current date in the 'inventory entry date' field with no option to edit
  - Set sales price=125% of KBB value
- After user populates all the provided fields and presses the ***Save*** button.
  - Save customer data to the customer table only if customer is a new customer
  - Save vehicle data to vehicle table

```
--sales_price=1.25*kbb_value

INSERT INTO vehicle
  VALUES('{vin}', '{manufacturer_name}', '{vehicle_type}',
'{model_year}', '{model_name}', '{mileage}',
```

```
'{vehicle_condition}', '{vehicle_description}',
'{sales_price}', '{kbb_value}');
--query will be dynamically generated based on the number of
colors selected
INSERT INTO vehicle_color
  VALUES ('{vin}', '{color_1}'),...,('{vin}', '{color_n}');
```

```
INSERT INTO purchase
  (vin, customer_id, login_username)
VALUES
  ('{vin}', '{customer_id}', '{login_username}');
```

Sell Vehicle

Abstract Code:

- First populate the customer details following the actions in Search/add customer task
- On the Sales order form, provide a field to enter the *sales date*
    - User enters the *sales date* and presses the ***Enter*** button
        - Update the vehicle record with the 'sales date' in the vehicle table

```
INSERT INTO sale
  (vin, customer_id, login_username)
VALUES
  ('{vin}', '{customer_id}', '{login_username}');
```

Search/Add Vendor

Abstract Code:

- Provide a text field to the user to input a *vendor name*
    - User inputs a String in the input field and presses the ***Enter*** button
        - Display the vendors that match the criteria vendor.vendor_name == vendor_name. The following fields are displayed: Vendor name, phone number, address (street, city, state and postal code)

```
SELECT
```

```
  vendor_name,
  vendor_phone_number,
  street,
  city,
  state,
  postal_code
FROM vendor
WHERE
  vendor_name='{vendor_name}';
```

- If user selects a vendor from the output list, then fields in the parent form (Add Repair form) get populated with the selected vendor
- Also, provide an add vendor fields for adding a new vendor
    - Vendor name
    - Phone number
    - Street
    - City
    - State
    - Postal code
- Insert new user inputs into vendor table.

```
INSERT INTO vendor
  VALUES('{vendor_name}', '{vendor_phone_number}',
  '{street}', '{city}', '{state}', '{postal_code}')
```

Add Repair

Abstract Code:

- First populate the vendor details following the actions in Search/add vendor task
- Find all applicable recalls by using the nhtsa_recall_number to filter the recall table

```
SELECT
  recall.nhtsa_recall_number,
  recall.manufacturer_name,
  recall.recall_description
FROM recall
WHERE
  recall.nhtsa_recall_number='{nhtsa_recall_number}';
```

- Populate Add Repair form with the following fields: vin, repair_start_date, repair_end_date, vendor_name, nhtsa_recall_number, total_cost, repair_description, vehicle_status
    - User enters the repair information in the Add Repair form and presses the ***Enter*** button
        - Update the repair table with the user input from the Add Repair form.

```
INSERT INTO repair
  VALUES('{vin}', '{repair_start_date}', '{repair_end_date}',
'{vendor_name}', '{nhtsa_recall_number}', '{total_cost}',
'{repair_description}', 'PENDING');
```

View Vehicle Detail

Abstract Code:

- The vehicle detail page is reached via the Search page
    - Pull information for a vehicle where vehicle.vin == *vin* for the selected vehicle
- Define Public Facing View Vehicle as:
    - For the selected vehicle get vehicle_type, model_year, manufacturer, color, mileage, sales_price, and the description of the car from the vehicle_universe view

```
SELECT
  vehicle_type,
  model_year,
  manufacturer_name,
  vehicle_color_grouped.color,
  mileage,
  sales_price,
  vehicle_description
FROM vehicle
LEFT JOIN (
  SELECT
    vin,
    GROUP_CONCAT(color ORDER BY color ASC SEPARATOR ',') as
color
  FROM vehicle_color
  GROUP BY vin
) as vehicle_color_grouped
ON vehicle_color_grouped.vin=vehicle.vin
WHERE
```

```
  vehicle.vin='{vin}';
```

- If authenticated user == 'Manager':
    - Show Public Facing View Vehicle page
    - Additional fields are populated in the view including:
        - Seller Information
    - For the selected vehicle get the seller and buyer information from the customer table except their driver's license or tax ID number
        - Name (first and last) of the inventory clerk that purchased the car
        - Total cost of repairs, and repairs section listing details for all repairs just like would be shown on an inventory clerk's view.
        - Buyer's contact information (everything except their driver's license or tax ID number)

```
-- this query provides the inventory clerk and the seller
information

SELECT
  vehicle.vin,
  vehicle.vehicle_type,
  vehicle.model_year,
  vehicle.manufacturer_name,
  vehicle_color_grouped.color,
  vehicle.mileage,
  vehicle.sales_price,
  vehicle.vehicle_description,
  DATE_FORMAT(purchase.purchase_date, '%Y-%m-%d'),
  customer_consolidated.customer_id,
  customer_consolidated.phone_number,
  customer_consolidated.email,
  customer_consolidated.street,
  customer_consolidated.city,
  customer_consolidated.state,
  customer_consolidated.postal_code,
  user.user_first_name AS inventory_clerk_first_name,
  user.user_last_name AS inventory_clerk_last_name,
  customer_consolidated.business_name,
  customer_consolidated.pc_name,
  customer_consolidated.pc_title,
  customer_consolidated.customer_type,
  vehicle.kbb_value
```

```
FROM vehicle
LEFT JOIN (
  SELECT
    vin,
    GROUP_CONCAT(color ORDER BY color ASC SEPARATOR ',') as
color
  FROM vehicle_color
  GROUP BY vin
) as vehicle_color_grouped
ON vehicle_color_grouped.vin=vehicle.vin
LEFT JOIN purchase
ON purchase.vin=vehicle.vin
LEFT JOIN user
ON user.login_username=purchase.login_username
LEFT JOIN (
SELECT
  customer.customer_id,
  customer.phone_number,
  customer.email,
  customer.street,
  customer.city,
  customer.state,
  customer.postal_code,
  individual.driver_license_number,
  individual.individual_first_name,
  individual.individual_last_name,
  business.business_name,
  business.pc_name,
  business.pc_title,
  business.tax_id_number,
  CASE WHEN individual.driver_license_number IS NULL THEN
'Individual' ELSE 'Business' END AS customer_type
FROM customer
LEFT JOIN individual
ON customer.customer_id=individual.customer_id
LEFT JOIN business
ON customer.customer_id=business.customer_id
) as customer_consolidated
ON customer_consolidated.customer_id=purchase.customer_id
WHERE vehicle.vin='{vin}';

-- this query provides the salesperson and the buyer
information
SELECT
  vehicle.vin,
  vehicle.vehicle_type,
```

```
  vehicle.model_year,
  vehicle.manufacturer_name,
  vehicle_color_grouped.color,
  vehicle.mileage,
  vehicle.sales_price,
  vehicle.vehicle_description,
  DATE_FORMAT(sale.sales_date, '%Y-%m-%d'),
  customer_consolidated.customer_id,
  customer_consolidated.phone_number,
  customer_consolidated.email,
  customer_consolidated.street,
  customer_consolidated.city,
  customer_consolidated.state,
  customer_consolidated.postal_code,
  user.user_first_name AS inventory_clerk_first_name,
  user.user_last_name AS inventory_clerk_last_name,
  customer_consolidated.business_name,
  customer_consolidated.pc_name,
  customer_consolidated.pc_title,
  customer_consolidated.customer_type,
  vehicle.kbb_value
FROM vehicle
LEFT JOIN (
  SELECT
    vin,
    GROUP_CONCAT(color ORDER BY color ASC SEPARATOR ',') as
color
  FROM vehicle_color
  GROUP BY vin
) as vehicle_color_grouped
ON vehicle_color_grouped.vin=vehicle.vin
LEFT JOIN sale
ON sale.vin=vehicle.vin
LEFT JOIN user
ON user.login_username=sale.login_username
LEFT JOIN (
SELECT
  customer.customer_id,
  customer.phone_number,
  customer.email,
  customer.street,
  customer.city,
  customer.state,
  customer.postal_code,
  individual.driver_license_number,
  individual.individual_first_name,
```

```
  individual.individual_last_name,
  business.business_name,
  business.pc_name,
  business.pc_title,
  business.tax_id_number,
  CASE WHEN individual.driver_license_number IS NULL THEN
'Individual' ELSE 'Business' END AS customer_type
FROM customer
LEFT JOIN individual
ON customer.customer_id=individual.customer_id
LEFT JOIN business
ON customer.customer_id=business.customer_id
) as customer_consolidated
ON customer_consolidated.customer_id=sale.customer_id
WHERE vehicle.vin='{vin}';
```

- Repairs Information
  - For the selected vehicle get all the repairs information from the repair table
    - Total for all repair costs
    - Details for all repairs: vendor, start date, end date, status, cost, and the recall number, if applicable

```
--repairs view
SELECT
  repair.vin,
  repair.repair_start_date,
  repair.repair_end_date,
  repair.vendor_name,
  repair.nhtsa_recall_number,
  repair.total_cost,
  repair.repair_description,
  repair.repair_status
FROM repair
WHERE
  vin='{vin}';
```

- Vehicle Information
- For the selected vehicle get original purchase price and purchase from the vehicle table (covered in above SQL query)
  - Original purchase price
  - Purchase date

        - Sales date
    - User Information (covered in above SQL query)
        - For the selected vehicle get the sales person's name from the user table
            - Salesperson's name (first and last)
- If authenticated User == 'Salespeople':
    - Show Public Facing View Vehicle page
    - Additional link is displayed to sell the car, ***sell vehicle*** , and link to the Sales Order form
    - If User clicks on ***sell vehicle*** link:
        - Load Sales Order form
- If authenticated User == 'Inventory Clerk':
    - Show Public Facing View Vehicle page
    - Additional fields are populated in the view including:
        - Vehicle Information
            - For the selected vehicle get original purchase price and purchase from the vehicle table
                - Original purchase price
        - Repairs Information (use the same SQL query as manager)
            - Show the same information as a Manager's view for the repair section
    - There should also be a popup to display the repair description when user clicks the ***repair description*** button against each repair record
    - A button for ***update repair*** should also be displayed and when clicked the Update Repair form will display
        - User will invoke the update repair task
    - A link for ***add repair*** should also be displayed and when clicked the Add Repair form will display
- If authenticated User == ' Owner ' (owner's query will display the aggregated information for all types of user roles)
    - Show Public Facing View Vehicle page
    - Additional fields that are a union of the view for Managers, Salespeople, and Inventory Clerk will display
    - If vehicle displayed in details page is sold:
        - Disable ***add repair*** link
        - Disable ***update repair*** button
        - Disable ***sell vehicle*** link

Update Repair

Abstract Code:

- For the selected vehicle and selected repair:
  - User is presented with a drop down for the repair status:
    - Dropdown is populated with 3 options: 'Pending', 'In Progress', and 'Complete'
    - The default selection on the dropdown will be the current status of the vehicle
    - User can update the status to 'In Progress' or 'Complete'
    - User clicks ***save*** button to save the status of the repair and the status is written to the repair table
  - If the updated status == 'Complete':
    - Find the corresponding vehicle, get the current sales price and change it to current sales price + 110% of the total cost of the repair

```
UPDATE repair
SET
  repair_status='{repair_status}'
WHERE
  vin='{vin}' AND
  repair_start_date='{start_date}';

--- if updated status = Complete, the sales_price for the
vehicle needs to be updated.

-- Get purchase price and save to variable: '{purchase_price}'
SELECT
  kbb_value AS purchase_price
FROM vehicle
WHERE vin='{vin}';

-- Get total repair cost and save to variable:
'{total_repair_cost}'
SELECT
  sum(total_cost) AS total_repair_cost
FROM repair
WHERE vin='{vin}';

-- Update vehicle sales_price
UPDATE vehicle
```

```
SET
  vehicle.sales_price='{purchase_price}' +
1.25*'{total_repair_cost}'
WHERE vin='{vin}';
```

View Reports

Abstract Code:

- If User.Role == 'Manager' OR 'Owner' :
    - User is able to click reports link from the search page and is presented with dropdown of all the report types
    - If a report is selected from the dropdown the selected report will be displayed
    - Seller History Report
        - Find all purchased vehicles from the vehicle table
        - For each vehicle sold get the name of the seller from the customer table
            - For each seller find all the vehicles sold:
                - Count vehicles as total number of vehicles
                - Take average purchase price
                - For each vehicle find all the repairs and sum the number of repairs
                - Average number of repairs is calculated as sum of all repairs for a seller / count of vehicles sold by them
        - Display all metrics and sort by total number of vehicles sold in descending order followed by average purchase price ascending
        - If the average number of repairs >= 5 highlight that seller in red

```
SELECT
  customer_name,
  COUNT(*) AS vehicles_sold,
  AVG(purchase_price) AS avg_purchase_price,
  AVG(number_of_repairs) AS avg_number_of_repairs
FROM
(SELECT
  individual.individual_first_name + " " +
individual.individual_last_name AS customer_name,
  vehicle.vin,
  a.number_of_repairs,
  vehicle.kbb_value AS purchase_price
FROM vehicle
```

```
LEFT JOIN purchase
ON vehicle.vin=purchase.vin
LEFT JOIN customer
ON purchase.customer_id=customer.customer_id
LEFT JOIN individual
ON customer.customer_id=individual.customer_id
LEFT JOIN
(SELECT
  vin,
  COUNT(1) AS number_of_repairs
FROM repair
GROUP BY vin) a
ON vehicle.vin=a.vin
WHERE
  individual.individual_first_name IS NOT NULL -- only
individuals

UNION ALL

SELECT
  business.business_name AS customer_name,
  vehicle.vin,
  a.number_of_repairs,
  vehicle.kbb_value AS purchase_price
FROM vehicle
LEFT JOIN purchase
ON vehicle.vin=purchase.vin
LEFT JOIN customer
ON purchase.customer_id=customer.customer_id
LEFT JOIN business
ON customer.customer_id=business.customer_id
LEFT JOIN
(SELECT
  vin,
  COUNT(1) AS number_of_repairs
FROM repair
GROUP BY vin) a
ON vehicle.vin=a.vin
WHERE
  business.business_name IS NOT NULL) b
GROUP BY customer_name
ORDER BY
  vehicles_sold DESC,
  avg_purchase_price ASC;
```

- Inventory Age Report

- Find all purchased vehicles not sold from the vehicle table
- By vehicle type for every vehicle:
  - Calculate age of vehicle as current date - purchased date in days
  - Get minimum of age
  - Get maximum of age
  - Get average of age
- Display all metrics for Inventory Age Report

```
-- displaying N/A when no data exists will be handled in UI
SELECT
  vehicle_type.vehicle_type,
  COALESCE(a.min_age, 'N/A') AS min_age,
  COALESCE(a.avg_age, 'N/A') AS avg_age,
  COALESCE(a.max_age, 'N/A') AS max_age
FROM vehicle_type
LEFT JOIN
(SELECT
  vehicle_type,
  MIN(DATEDIFF(CURRENT_TIMESTAMP, purchase.purchase_date)) AS
min_age,
  AVG(DATEDIFF(CURRENT_TIMESTAMP, purchase.purchase_date)) AS
avg_age,
  MAX(DATEDIFF(CURRENT_TIMESTAMP, purchase.purchase_date)) AS
max_age
FROM vehicle
LEFT JOIN purchase
ON vehicle.vin=purchase.vin
LEFT JOIN sale
ON vehicle.vin=sale.vin
WHERE
  sale.sales_date IS NULL
GROUP BY vehicle_type) AS a
on vehicle_type.vehicle_type=a.vehicle_type;
```

- Average Time in Inventory Report
  - Find all sold and purchased vehicles from the vehicle table
  - By vehicle type for every vehicle:
    - Calculate age of vehicle as sales date - purchased date in days as time in inventory
    - Get an average of time in inventory
  - Display all metrics for Average Time in Inventory Report

```
-- displaying N/A when no data exists will be handled in UI
```

```
SELECT
  vehicle_type.vehicle_type,
  COALESCE(a.min_age, 'N/A') AS min_age,
  COALESCE(a.avg_age, 'N/A') AS avg_age,
  COALESCE(a.max_age, 'N/A') AS max_age
FROM vehicle_type
LEFT JOIN
(SELECT
  vehicle_type,
  MIN(DATEDIFF(sales_date, purchase_date)) AS min_age,
  AVG(DATEDIFF(sales_date, purchase_date)) AS avg_age,
  MAX(DATEDIFF(sales_date, purchase_date)) AS max_age
FROM vehicle
LEFT JOIN purchase
ON vehicle.vin=purchase.vin
LEFT JOIN sale
ON vehicle.vin=sale.vin
WHERE
  sales_date IS NOT NULL -- only for sold vehicles
GROUP BY vehicle_type) AS a
ON a.vehicle_type=vehicle_type.vehicle_type;
```

- Price Per Condition Report
  - Find all vehicles from the vehicle table
  - By vehicle type and condition (Excellent, Very Good, Good, Fair) for every vehicle:
    - Get average price paid
  - Display all metrics for Price Per Condition Report and pivot on vehicle type displayed as rows and condition as columns

```
SELECT
  vehicle_type,
  AVG(CASE WHEN vehicle_condition='Excellent' THEN kbb_value
ELSE 0 END) AS 'Condition=Excellent',
  AVG(CASE WHEN vehicle_condition='Very Good' THEN kbb_value
ELSE 0 END) AS 'Condition=Very Good',
  AVG(CASE WHEN vehicle_condition='Good' THEN kbb_value ELSE 0
END) AS 'Condition=Good',
  AVG(CASE WHEN vehicle_condition='Fair' THEN kbb_value ELSE 0
END) AS 'Condition=Fair'
FROM vehicle
```

```
GROUP BY vehicle_type;
```

- Repair Statistics Report
  - Find all vendor names from vendor table
  - Find all repairs from repair table
  - For each vendor get all repairs:
    - Count the repairs that have a status == 'Complete'
    - Sum the repair cost as total dollar amount spent on completed repairs
    - Get average number of repairs per vehicle
    - Calculate age as end date - start date in days as length of time to complete repair
      - Get average length of time to complete repair
  - Display all metrics for Repair Statistics Report

```
-- number_of_repairs / number_of_vehicles = average number of
repairs per vehicle. This will be done in the UI.

SELECT
  vendor_name,
  COUNT(*) AS number_of_repairs,
  SUM(total_cost) AS total_spend,
  COUNT(distinct vin) AS number_of_vehicles,
  AVG(DATEDIFF(repair_end_date, repair_start_date)) AS
avg_duration
FROM repair
WHERE
  repair_status='Complete'
GROUP BY vendor_name;
```

- Monthly Sales Report
  - Find all sales dates, and sales price, purchase price from vehicle table
  - Find all repair costs from repair table
  - Find salespeople from users table
  - From sales, repairs, and salespeople:
    - By year, month, and salesperson:
      - Count all sales dates as total number of vehicles sold
      - Sum sales price as total sales income
      - Sum purchase costs as total purchase cost

        - Sum repair costs as total repair cost
        - Calculate total net income as total sales income - total repair cost - total purchase cost
    - Display all metrics for Monthly Sales Report by year and month aggregation and sort in descending order
    - Drilldown on year and month from results is available as a second display of this report
        - Display all metrics for Monthly Sales Report by year, month, and salesperson ordered by top performing salespeople in descending order (in the event of a tie, the salesperson who has sold the highest dollar value will be considered the top salesperson)

```
-- Part 1: Summary report
SELECT
  DATE_FORMAT(sales_date, '%Y-%m') as ym,
  COUNT(1) AS number_of_vehicles,
  SUM(sales_price) AS sales_revenue,
  SUM(sales_price - repair_cost - kbb_value) AS net_income
FROM
(SELECT
 vehicle.vin,
 sale.sales_date,
 vehicle.sales_price,
 vehicle.kbb_value,
 repair_total_cost_by_vin.total_cost AS repair_cost
FROM vehicle
LEFT JOIN sale
ON vehicle.vin=sale.vin
LEFT JOIN (
SELECT
  repair.vin,
  SUM(repair.total_cost) AS total_cost
FROM repair
GROUP BY vin
) as repair_total_cost_by_vin
ON repair_total_cost_by_vin.vin=vehicle.vin
WHERE
  sale.sales_date IS NOT NULL) as a  -- vehicle must be sold
GROUP BY ym
order by ym DESC;
```

```
-- Part 2: Drilldown report
-- ym is the selected year-month combination
SELECT
  user.user_first_name,
  user.user_last_name,
  COUNT(1) AS number_of_vehicles,
  SUM(vehicle.sales_price) AS total_sales
FROM vehicle
LEFT JOIN sale
ON vehicle.vin=sale.vin
LEFT JOIN user
ON sale.login_username=user.login_username
WHERE
  DATE_FORMAT(sales_date, '%Y-%m')='{ym}'
GROUP BY
 user.login_username
ORDER BY
 number_of_vehicles DESC,
 total_sales DESC;
```