Table of Contents

<u>Login</u>

Abstract Code

- User enters *username* ('$Username'), *password* ('$Password') input fields.
- If data validation is successful for both *username* and *password* input fields, then:
  - When ***Enter*** button is clicked:

```
SELECT password FROM LoginUser WHERE username= '$Username';
```

    - If User record is found but user.password != '$Password':
      - Go back to **<u>Login</u>** form, with error message.
    - Else:
      - Store login information as session variable '$UserName'.
- Else *username* and *password* input fields are invalid, display **<u>Login</u>** form, with error message.

Add Customer

Abstract code:

● Upon clicking the ***Add Customer*** button:

● Add the customer using driver license or taxID

● Store the customer information as variable '$Customer', enter the *address*, *phone* and *email* (if any) input fields

```
INSERT INTO Customer (customerID, email, phone, street_address, city, state,
postal_code)
VALUES ('$customerID', '$email', '$phone', '$street_address', '$city', '$state',
'$postal_code');
```

● If it is an individual, enter *driver license*, the *first name* and *last name* input fields

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

● Else if it is a business, enter *taxID*, the *business name*, primary *contact name* and *title* input fields

```
INSERT INTO Business (taxID, business_name, title, contact_name, customerID)
VALUES ('$TaxID', '$business_name', '$title', '$contact_name', '$customerID');
```

Lookup Customer

Abstract code:

- Upon clicking the ***Lookup Customer*** button:
- Find the customer using driver license if it is an individual or taxID if business;
    - If no result found:
        - Display error message: “Can’t find it!”
    - Else:
        - Display the address, phone and email (if not null)
        - If it is an individual, find the individual and display the first and last name

```
SELECT email, phone, address, first_name, last_name, customerID
FROM Customer AS C INNER JOIN Individual AS I ON C.customerID = I.customerID
WHERE driver_license= '$driver_license';
```

- Else if it is a business, find the business and display the business name, primary contact name and title.

```
SELECT email, phone, address, business_name, contact_name, title, customerID
FROM Customer AS C INNER JOIN Business AS B ON C. customerID = B.customerID
WHERE taxID= '$TaxID';
```

Add Vehicle

Abstract code

- Clerk clicks the ***Add Vehicle*** button from the page after they logged in—Jump to the **new vehicle** form
- Run the **Add Vehicle** task
    - Add a unique alphanumeric Vehicle Identification Number (VIN), model name, model year, manufacturer info, invoice price, is_sold status, add date, Inventory clerk's username to the Vehicle table
    - If there is additional information:
      Add the description

```
INSERT INTO Vehicle (VIN, model_name, model_year, M_ID, invoice_price, is_sold,
add_date, `description`, username)
VALUES ('$VIN', '$model_name', '$model_year', '$M_ID', '$Invoice_price', '$is_sold',
'$add_date', '$description', '$username');
```

If vehicle type is car:

```
INSERT INTO Car (VIN, `type`, door_num) VALUES ('$VIN', '$type', '$DoorNum');
```

If vehicle type is convertible:

```
INSERT INTO Convertible (VIN, `type`, roof_type, backseat_num) VALUES ('$VIN',
'$type', '$roof_type', '$backseat_num');
```

If vehicle type is truck:

```
INSERT INTO Truck (VIN, `type`, cargo_capacity, cargocover_type, rear_axle_num)
VALUES ('$VIN', '$type', '$cargo_capacity', '$cargocover_type', '$rear_axle_num');
```

If vehicle type is VanMinivan:

```
INSERT INTO VanMinivan (VIN, `type`, driverside_backdoor)
VALUES ('$VIN', '$type', '$driverside_backdoor');
```

If vehicle type is SUV:

```
INSERT INTO SUV (VIN, `type`, drivetrain_type, cupholder_num)
VALUES ('$VIN', '$type', '$drivetrain_type', '$cupholder_num');
```

- Add color(s) with color names from given list:
  If a vehicle has multiple colors:

For each color:

```
INSERT INTO VehicleColor (VIN, color)
VALUES ('$VIN', '$Color');
```

End for

- Click the ***Save*** button;
    - If all inputs are valid data type; Go to the vehicle detail page
        - Else:
          Display error message: "Input invalid!"

<u>Search Vehicle</u>

Abstract Code:

- Display initial search page with total number of vehicles available for purchase in the system
- Run the **Search Vehicle** task
    - User selects from the dropdown box of the Vehicle type, Manufacturer, Model year, and Color field
    - User enters a price range for the list price field
    - User enters the entire or substring of the Manufacturer, Model year, Model name or description in the keyword filed
    - Upon clicking the ***Search*** button:
        - If all inputs are valid data type; Display vehicles with that match all entered fields
            - Sort the results by VIN in ascending order
            - Display the following attributes for each vehicle that matches: Vehicle type, Model Year, Manufacturer, Model name, Color(s), List Price
            - If a vehicle has multiple colors:
              return a single row with all colors listed

```
WITH VehicleWithType (VIN, model_name, model_year, list_price, vehicle_type) AS
(SELECT V.VIN, model_name, model_year, 1.25 * invoive_price AS list_price,
COALESCE(Car.type, Con.type, Truck.type, VM.type, SUV.type) as vehicle_type
FROM Vehicle AS V
LEFT JOIN Car ON V.VIN = Car.VIN
LEFT JOIN Convertible AS Con ON V.VIN = Con.VIN
LEFT JOIN Truck ON V.VIN = Truck.VIN
LEFT JOIN VanMinivan AS VM ON V.VIN = VM.VIN
LEFT JOIN SUV ON V.VIN = SUV.VIN
)
, VehicleColorList(VIN, colorlist) AS
(SELECT V.VIN, GROUP_CONCAT(color) FROM Vehicle AS V
INNER JOIN VehicleColor AS VC on V.VIN = VC.VIN
GROUP BY V.VIN
)
SELECT VWT.VIN, model_name, model_year, list_price, vehicle_type, M.name,
VCL.colorlist FROM VehicleWithType AS VWT
INNER JOIN Manufacturer AS M on VWT.M_ID = M.M_ID
INNER JOIN VehicleColorList AS VCL on VWT.VIN = VCL.VIN
WHERE is_sold = FALSE AND
CASE WHEN '$vehicle_type' IS NOT NULL THEN vehicle_type = '$vehicle_type'
ELSE vehicle_type = vehicle_type
END
CASE WHEN '$model_year' IS NOT NULL THEN model_year = '$model_year' ELSE
model_year = model_year
END
CASE WHEN '$color' IS NOT NULL THEN color = '$color' ELSE color = color
```

```
END
CASE WHEN '$min_price' IS NOT NULL THEN price >= '$min_price' ELSE price >= 0
END
CASE WHEN '$max_price' IS NOT NULL THEN price <= '$max_price' ELSE price <=
binary_double_infinity
END
CASE WHEN '$substring' IS NOT NULL THEN M.name contains '$substring' or
model_year contains '$substring' or model_name contains '$substring' or description
contains '$substring' ELSE 1=1
END
ORDER BY VIN ASC;
```

- If user is login_user:
  User enters a VIN in the VIN field in addition to the above choices
  Display the invoice price

```
WITH VehicleWithType (VIN, model_name, model_year, invoice_price, list_price,
vehicle_type,) AS
(SELECT V.VIN, model_name, model_year, invoice_price, 1.25 * invoive_price AS
list_price, COALESCE(Car.`type`, Con.`type`, Truck.`type`, VM.`type`, SUV.`type`) as
vehicle_type
FROM Vehicle AS V
LEFT JOIN Car ON V.VIN = Car.VIN
LEFT JOIN Convertible AS Con ON V.VIN = Con.VIN
LEFT JOIN Truck ON V.VIN = Truck.VIN
LEFT JOIN VanMinivan AS VM ON V.VIN = VM.VIN
LEFT JOIN SUV ON V.VIN = SUV.VIN
)

, VehicleColorList(VIN, colorlist) AS
(SELECT V.VIN, GROUP_CONCAT(color) FROM Vehicle AS V
INNER JOIN VehicleColor AS VC on V.VIN = VC.VIN
GROUP BY V.VIN
)
SELECT V.VIN, model_name, model_year, invoice_price, list_price, vehicle_type,
M.`name`, VCL.colorlist FROM VehicleWithType AS VWT
INNER JOIN Manufacturer AS M on VWT.M_ID = M.M_ID
INNER JOIN VehicleColorList AS VC on VWT.VIN = VCL.VIN
WHERE
CASE WHEN '$VIN'  IS NOT NULL THEN VIN = '$VIN'  ELSE VIN = VIN
END
CASE WHEN '$is_sold' IS NOT NULL THEN is_sold = '$Is_sold ' ELSE is_sold =
is_sold
END
CASE WHEN '$vehicle_type' IS NOT NULL THEN vehicle_type = '$vehicle_type' ELSE
vehicle_type = vehicle_type
```

```
END
CASE WHEN '$model_year' IS NOT NULL THEN model_year = '$model_year' ELSE
model_year = model_year
END
CASE WHEN '$color' IS NOT NULL THEN color = '$color' ELSE color = color
END
CASE WHEN '$min_price' IS NOT NULL THEN price >= '$min_price' ELSE price >= 0
END
CASE WHEN '$max_price' IS NOT NULL THEN price <= '$max_price' ELSE price <=
binary_double_infinity
END
CASE WHEN '$substring' IS NOT NULL THEN M.name contains '$substring' or
model_year contains '$substring' or model_name contains '$substring' or description
contains '$substring' ELSE 1=1
END
ORDER BY VIN ASC;
```

- If no result is found:
  Display “Sorry, it looks like we don’t have that in stock!”
- Else:
  When click an individual result, go to the vehicle detail page;
  Run the **Get Vehicle Detail** task

- Else:
  Display error message: “Invalid input!”
  Go back to search page

Get Vehicle Detail

Abstract code:

When click an individual vehicle search result:

- Display the following attributes for each vehicle that matches: VIN, Vehicle type, attributes to that vehicle type, Model Year, Manufacturer, Model name, Color(s), List Price and the description for the selected result

If an individual Car is chosen:

```
WITH VehicleColorList(VIN, colorlist) AS
(SELECT V.VIN, GROUP_CONCAT(color) FROM Vehicle AS V
INNER JOIN VehicleColor AS VC on V.VIN = VC.VIN
GROUP BY V.VIN
)
SELECT V.VIN, Car.type, M.name, model_year, VCL.colorlist, invoice_price * 1.25
AS list_price, door_num FROM Vehicle AS V
INNER JOIN VehicleColorList AS VCL ON V.VIN = VCL.VIN
INNER JOIN Manufacturer as M ON V.M_ID = M.M_ID
INNER JOIN Car ON V.VIN = Car.VIN
Where VIN = '$VIN';
```

If an individual Truck is chosen:

```
WITH VehicleColorList(VIN, colorlist) AS
(SELECT V.VIN, GROUP_CONCAT(color) FROM Vehicle AS V
INNER JOIN VehicleColor AS VC on V.VIN = VC.VIN
GROUP BY V.VIN
)
SELECT V.VIN, Truck.type, M.name, model_year, VCL.colorlist, invoice_price * 1.25
AS list_price, cargo_capacity, cargoc ver_type, rear_axle_num FROM Vehicle AS V
INNER JOIN VehicleColorList AS VCL ON V.VIN = VCL.VIN
INNER JOIN Manufacturer AS M ON V.M_ID = M.M_ID
INNER JOIN Truck ON V.VIN = Truck.VIN
Where VIN = '$VIN';
```

If an individual Convertible is chosen:

```
WITH VehicleColorList(VIN, colorlist) AS
(SELECT V.VIN, GROUP_CONCAT(color) FROM Vehicle AS V
INNER JOIN VehicleColor AS VC on V.VIN = VC.VIN
GROUP BY V.VIN
)
SELECT V.VIN, CON.type, M.name, model_year, VCL.colorlist, invoice_price * 1.25
AS list_price, roof_type, backseat_num FROM Vehicle AS V
INNER JOIN VehicleColorList AS VCL ON V.VIN = VCL.VIN
INNER JOIN Manufacturer AS M ON V.M_ID = M.M_ID
INNER JOIN Convertible AS CON ON V.VIN = CON.VIN
Where VIN = '$VIN';
```

If an individual VanMinivan is chosen:

```
WITH VehicleColorList(VIN, colorlist) AS
(SELECT V.VIN, GROUP_CONCAT(color) FROM Vehicle AS V
INNER JOIN VehicleColor AS VC on V.VIN = VC.VIN
GROUP BY V.VIN
)
SELECT V.VIN, VanMinivan.type, M.name, model_year, VCL.colorlist, invoice_price
* 1.25 AS list_price, driveside_backdoor FROM Vehicle AS V
INNER JOIN VehicleColorList AS VCL ON V.VIN = VCL.VIN
INNER JOIN Manufacturer AS M ON V.M_ID = M.M_ID
INNER JOIN VanMinivan AS VM ON V.VIN = VM.VIN
Where VIN = '$VIN';
```

If an individual SUV is chosen:

```
WITH VehicleColorList(VIN, colorlist) AS
(SELECT V.VIN, GROUP_CONCAT(color) FROM Vehicle AS V
INNER JOIN VehicleColor AS VC on V.VIN = VC.VIN
GROUP BY V.VIN
)
SELECT V.VIN, SUV.type, M.name, model_year, VCL.colorlist, invoice_price * 1.25
AS list_price, drivetrain_type, cupholder_num FROM Vehicle AS V
INNER JOIN VehicleColorList AS VCL ON V.VIN = VCL.VIN
INNER JOIN Manufacturer AS M ON V.M_ID = M.M_ID
INNER JOIN SUV ON V.VIN = SUV.VIN
Where VIN = '$VIN';
```

- If User is Manager, display all the following information:
    - Display the inventory clerk name (first and last) that added the vehicle, the invoice price, and the date it was added to inventory.

```
SELECT VIN, invoice_price, add_date, LU.first_name, LU.last_name FROM Vehicle
AS V
INNER JOIN LoginUser AS LU ON V.username = LU.username
```

- if it has been sold, display the buyer’s contact information (everything except their driver’s license or taxID number), list price, sold price, sales date, and the salesperson’s name (first and last)

First, find out the customer type using ‘$customerID’.

```
SELECT (CASE WHEN '$customerID' IN (SELECT customerID FROM Individual)
            THEN 'Individual'
                ELSE 'Business'
                END AS customer_type
WHERE 1 = 1
```

If the customer type is ‘Individual’, then find the sale information including list price, sold price, sold date, and saleperson’s name, and find the customer’s information including customer name, email, phones, street address, city, state, postal code.

```
WITH SaleInfo (username, VIN, customerID, list_price, sold_price, sold_date,
saleperson_name) AS
(SELECT Sale.username, Sale.VIN, Sale.customerID,
            Vehicle.invoice_price * 1.25 AS list_price,
            sold_price,
            sold_date,
            CONCAT(LU.first_name, LU.last_name) AS saleperson_name,
FROM Sale
          INNER JOIN LoginUser AS LU ON Sale.username = LU.username
          INNER JOIN Vehicle ON Sale.VIN = Vehicle.VIN
WHERE VIN = '$VIN'
)

SELECT CONCAT(I.first_name, I.last_name) as buyer_name,
            email, phone, street_address, city, state, postal_code,
            invoice_price * 1.25 as list_price,
            sold_price,
            sold_date,
            saleperson_name
FROM SaleInfo AS SI
          INNER JOIN Customer AS C ON SI.customerID = C.customerID
          INNER JOIN Individual AS I ON SI.customerID = I.customerID
```

If the customer type is ‘Business, then find the sale information including list price, sold price, sold date, and saleperson’s name, and find the customer’s information including business name, email, phones, street address, city, state, postal code, title, and contact name.

```
WITH SaleInfo (username, VIN, customerID, list_price, sold_price, sold_date,
saleperson_name) AS
(SELECT Sale.username, Sale.VIN, Sale.customerID,
            Vehicle.invoice_price * 1.25 AS list_price,
            sold_price,
            sold_date,
            CONCAT(LU.first_name, LU.last_name) AS saleperson_name,
FROM Sale
          INNER JOIN LoginUser AS LU ON Sale.username = LU.username
          INNER JOIN Vehicle ON Sale.VIN = Vehicle.VIN
WHERE VIN = '$VIN'
)
```

```
SELECT business_name as buyer_name,
            email, phone, street_address, city, state, postal_code,
            title, contact_name,
            invoice_price * 1.25 as list_price,
            sold_price,
            sold_date,
            saleperson_name
FROM SaleInfo AS SI
          INNER JOIN Customer AS C ON SI.customerID = C.customerID
          INNER JOIN Business AS B ON SI.customerID = B.customerID
```

- If any repairs have been made for the vehicle, display the customer name (first and last name for individuals, or company name for companies), the service writer who entered the repair, and the repair's start date, end date, labor charges, parts cost, and total cost.

```
WITH CustomerName(customerID, customer_name) AS
(SELECT
            CASE WHEN (driver_license is NOT Null) THEN (I.customerID)
            ELSE (B.customerID)
            END AS (customerID),

      CASE WHEN (driver_license is NOT Null) THEN (CONCAT(first_name,
last_name))
            ELSE (B.name)
            END AS (customer_name)
FROM Individual AS I FULL OUTER JOIN Business AS B ON I.customerID =
B.customerID)


,PartsCostACustomer (VIN, customerID, start_date, all_parts_cost) AS
(SELECT VIN, customerID, start_date, sum(part_price * part_quantity)
FROM Part
WHERE customerID == '$customerID'
GROUP BY VIN, customerID, start_date)

SELECT customer_name,
            CONCAT(LU.first_name, ' ', LU.last_name) as servicewriter_name,
            start_date, end_date, labor_cost, all_parts_cost,
      (all_parts_cost + larbor_cost) as total_cost
FROM Repair AS R
          INNER JOIN PartsCostACustomer AS PCC ON R.VIN = PCC.VIN AND R.VIN
= PCC.customerID AND R.start_date = PCC.start_date
```

```
        INNER JOIN CustomerName AS CN ON Repair.customerID = CN.customerID
        INNER JOIN Vehicle AS V ON R.VIN = V.VIN
        INNER JOIN LoginUser AS LU ON Vehicle.username = LU.username;
```

Add Sale

Abstract Code

- Find the vehicle using VIN by running the **Get Vehicle Detail** task:
    - If Vehicle is not found or Vehicle's IsSold == "True":
        - Display error message and go back to search page
    - Else:
        - Display the list price and invoice price, VIN, Vehicle type, Model Year, Manufacturer, Model, and Color(s)

```
WITH VehicleWithType (VIN, model_name, model_year, invoice_price, list_price,
vehicle_type,) AS
(SELECT V.VIN, model_name, model_year, invoice_price, 1.25 * invoive_price AS
list_price, COALESCE(Car.type, Con.type, Truck.type, VM.type, SUV.type) as
vehicle_type
FROM Vehicle AS V
LEFT JOIN Car ON V.VIN = Car.VIN
LEFT JOIN Convertible AS Con ON V.VIN = Con.VIN
LEFT JOIN Truck ON V.VIN = Truck.VIN
LEFT JOIN VanMinivan AS VM ON V.VIN = VM.VIN
LEFT JOIN SUV ON V.VIN = SUV.VIN
)

, VehicleColorList(VIN, colorlist) AS
(SELECT V.VIN, GROUP_CONCAT(color) FROM Vehicle AS V
INNER JOIN VehicleColor AS VC on V.VIN = VC.VIN
GROUP BY V.VIN
)
SELECT VIN, vehicle_type, model_year, model_name, list_price, invoice_price,
M.name, VCL.colorlist FROM VehicleWithType AS VWT
INNER JOIN Manufacturer AS M on VWT.M_ID = M.M_ID
INNER JOIN VehicleColorList AS VCL on VWT.VIN = VCL.VIN
WHERE VWT.VIN = '$VIN';
```

- Click the ***Add Sale*** button on the vehicle detail page- Jump to the **Sales Order** form
- Run the **Add Sale** task:
    - Add the sold price and sold date
        - If login_user is salesperson:
            - If sold price is less than or equal to 95% of the invoice price, go back to the ***Add Sale*** button with error message
    - Run the **Lookup Customer** task
        - If Customer record is not found:
          Run the **Add Customer** task

```
INSERT INTO Sale (username, VIN, customerID, sold_date, sold_price)
VALUES ('$username', '$VIN', '$customerID', '$sold_date', '$sold_price');
```

- Click the ***Save*** button; Go back to the vehicle detail page

View Sale

Abstract Code

● Manager or owner click on ***View Sale*** button from the page after they logged in.
● Run the **View Sale** task: query for information about the vehicle that was sold and the clerk who added the vehicle, the salesperson who and when added the sale and the sold price, and the customer who bought the vehicle.

- Find the vehicle that was sold using VIN; Display the list price and invoice price, VIN, Vehicle type, Model Year, Manufacturer, Model, and Color(s)
- Find the login_user using username; Display the first name and last name
- Find the clerk who added the vehicle using username; Display the date it was added
- Find the salesperson who added the sale transaction using username
- Find the sale; Display sold date and sold price
- Find the customer by running the **Lookup Customer** task

```
WITH SaleWithSaleperson (username, VIN, customerID, sold_date, sold_price,
saleperson_name) AS
(SELECT username, VIN, customerID, sold_date, sold_price
            CONCAT(first_name, last_name) AS saleperson_name
FROM Sale INNER JOIN LoginUser ON Sale.username = LoginUser.username
WHERE VIN = '$VIN'
)

,CustomerName(customerID, customer_name) AS
(SELECT
            CASE WHEN (driver_license is NOT Null) THEN (Individual.customerID)
            ELSE (Business.customerID)
            END AS (customerID),

      CASE WHEN (driver_license is NOT Null) THEN (CONCAT(first_name,
last_name))
            ELSE (Business.name)
            END AS (customer_name),
FROM Individual FULL OUTER JOIN Business ON Individual.customerID =
Business.customerID
)

,VehicleWithType (VIN, model_name, model_year, invoice_price, list_price,
vehicle_type,) AS
SELECT (V.VIN, model_name, model_year, invoice_price, 1.25 * invoive_price AS
list_price, COALESCE(Car.type, Con.type, Truck.type, VM.type, SUV.type) AS
vehicle_type

,VehicleColorList(VIN, colorlist) AS
(SELECT V.VIN, GROUP_CONCAT(color) FROM Vehicle AS V
INNER JOIN VehicleColor AS VC on V.VIN = VC.VIN
GROUP BY V.VIN)

FROM Vehicle AS V
LEFT JOIN Car ON V.VIN = Car.VIN
LEFT JOIN Convertible AS Con ON V.VIN = Con.VIN
LEFT JOIN Truck ON V.VIN = Truck.VIN
LEFT JOIN VanMinivan AS VM ON V.VIN = VM.VIN
LEFT JOIN SUV ON V.VIN = SUV.VIN
)

SELECT Vehicle.VIN as VIN, model_name, model_year, invoice_price, is_sold,
description, list_price, sold_date, VWT.vehicle_type, VCL.colorlist, Manufacturer.name
as manufacturer_name,
      LoginUser.username as inventoryClerk_name,
      Vehicle.add_date as add_date,
      CustomerName.customer_name as customer_name,
      SaleWithSaleperson.saleperson_name as saleperson_name,
FROM SaleWithSaleperson INNER JOIN Vehicle ON SaleWithSaleperson.VIN =
Vehicle.VIN
          INNER JOIN LoginUser ON Vehicle.username = LoginUser.username
          INNER JOIN CustomerName ON SaleWithSaleperson.customerID =
CustomerName.customerID
          INNER JOIN Manufacturer ON Vehicle.M_ID = Manufacturer.M_ID
          INNER JOIN VehicleWithType AS VWT ON Vehicle.VIN = VWT.VIN
          INNER JOIN VehicleColorList AS VCL ON Vehicle.VIN = VCL.VIN
```

<u>Add Repair</u>

Abstract Code:

- If ***New Repair*** button is clicked by user:
  - Find the customer by **Lookup Customer**
    - If not found Customer:
      add a new Customer by **Add Customer**
  - If ***Save*** button is clicked:
    - Check if info such as data type and range of Odometer reading and current date is entered correctly.
    - If the current date is different from other start dates:
      - add a <u>**Repair**</u> form to repair history.

```
INSERT INTO Repair (VIN, customerID, start_date, username, complete_date,
labor_charge, odometer, description)
VALUES ('$VIN', '$customerID', '$start_date', '$complete_date', '$username',
'$labor_charge', '$odometer', '$description');
```

```
INSERT INTO Part (VIN, customerID, start_date, username, partID, part_price,
part_quantity, vendor)
VALUES ('$VIN', '$customerID', '$start_date', '$complete_date', '$username', '$partID',
'$part_price', '$part_quantity');
```

- Else:
  - Display the error message

View Repair

● Upon clicking on ***View Repair*** button from the page after logged in—Jump to the Repair form

- Find the vehicle by VIN by running the **Search Vehicle** task
    - If Vehicle is not found or Vehicle's IsSold == "False":
        - Display error message and go back to Repair form
    - Else:
        - Display the list price and invoice price, VIN, Vehicle type, Model Year, Manufacturer, Model, and Color(s) to confirm it's the correct vehicle
        - Display the repair history including start date, end date, labor charges, parts cost, and total cost
        - Display an ***Edit*** button for each repair history
        - If all repairs are complete in repair history, display the ***New Repair*** button; Hide it otherwise;
- Find the login_user using username; Display the first name and last name
- Find the salesperson who entered the repair using username
- Find the customer by running the **Lookup Customer** task

```
WITH CustomerName(customerID, customer_name) AS
(SELECT
        CASE WHEN (driver_license is NOT Null) THEN (Individual.customerID)
        ELSE (Business.customerID)
        END AS (customerID),

    CASE WHEN (driver_license is NOT Null) THEN (CONCAT(first_name,
last_name))
        ELSE (Business.name)
        END AS (customer_name),
FROM Individual FULL OUTER JOIN Business ON Individual.customerID =
Business.customerID
)

, VehicleWithType(VIN, model_name, model_year, invoice_price, is_sold, description,
vehicle_type, M_ID, add_date, username) AS
(SELECT V.VIN, model_name, model_year, invocie_price, is_sold, description,
        COALESCE(Car.type, Con.type, Truck.type, VM.type, SUV.type) as
vehicle_type, M_ID, add_date, username
FROM Vehicle AS V
LEFT JOIN Car ON V.VIN = Car.VIN
LEFT JOIN Convertible AS Con ON V.VIN = Con.VIN
LEFT JOIN Truck ON V.VIN = Truck.VIN
LEFT JOIN VanMinivan AS VM ON V.VIN = VM.VIN
LEFT JOIN SUV ON V.VIN = SUV.VIN
)
```

```
,VehicleColorList(VIN, colorlist) AS
(SELECT V.VIN, GROUP_CONCAT(color) FROM Vehicle AS V
INNER JOIN VehicleColor AS VC on V.VIN = VC.VIN
GROUP BY V.VIN)

SELECT R.VIN, CN.customerID , R.start_date , complete_date, labor_charge,
odometer, description, username, VWT.vehicle_type, M.name, model_year,
VCL.colorlist, List_price, part_quantity,  sum(part_quantity * part_price),
sum(labor_charge) + sum(part_quantity * part_price)
FROM Repair AS R
INNER JOIN Vehicle AS V ON R.VIN = V.VIN
INNER JOIN VehicleWithType AS VWT ON V.VIN = VWT.VIN
INNER JOIN VehicleColorList AS C ON V.VIN = VCL.VIN
INNER JOIN Manufacturer as M ON V.M_ID = M.M_ID
INNER JOIN Part AS P ON R.VIN = P.VIN and R.customerID = P.customerID and
R.start_date = P.start_date and R.username = P.partID
INNER JOIN CustomerName AS CN ON R.customerID = CN.customerID
WHERE
CASE WHEN '$VIN' IS NOT NULL THEN VIN = '$VIN' ELSE VIN = VIN
END
CASE WHEN '$customerID' IS NOT NULL THEN customerID = '$cusomterID' ELSE
customerID = customerID
END;
```

Edit Repair

Abstract Code:

If ***Edit*** button is clicked by users:

- Display the corresponding **Repair** Form.

```
WITH CustomerName(customerID, customer_name) AS
(SELECT
CASE WHEN (driver_license is NOT Null) THEN (Individual.customerID)
ELSE (Business.customerID)
END AS (customerID),
CASE WHEN (driver_license is NOT Null) THEN (CONCAT(first_name, last_name))
ELSE (Business.name)
END AS (customer_name),
FROM Individual FULL OUTER JOIN Business ON Individual.customerID =
Business.customerID
)
, VehicleWithType(VIN, model_name, model_year, invoice_price, is_sold, description,
vehicle_type, M_ID, add_date, username) AS
(SELECT V.VIN, model_name, model_year, invocie_price, is_sold, description,
COALESCE(Car.type, Con.type, Truck.type, VM.type, SUV.type) as vehicle_type,
M_ID, add_date, username
FROM Vehicle AS V
LEFT JOIN Car ON V.VIN = Car.VIN
LEFT JOIN Convertible AS Con ON V.VIN = Con.VIN
LEFT JOIN Truck ON V.VIN = Truck.VIN
LEFT JOIN VanMinivan AS VM ON V.VIN = VM.VIN
LEFT JOIN SUV ON V.VIN = SUV.VIN

, VehicleColorList(VIN, colorlist) AS
(SELECT V.VIN, GROUP_CONCAT(color) FROM Vehicle AS V
INNER JOIN VehicleColor AS VC on V.VIN = VC.VIN
GROUP BY V.VIN
)

SELECT R.VIN, R.customerID , R.start_date , complete_date, labor_charge, odometer,
description, username, vehicle_type, M.name, model_year, VCL.colorlist, List_price,
part_quantity,  sum(part_quantity * part_price), sum(labor_charge) + sum(part_quantity
* part_price)
FROM Repair AS R
INNER JOIN Vehicle AS V ON R.VIN = V.VIN
INNER JOIN VehicleColorList AS VCL ON V.VIN = VCL.VIN
INNER JOIN Manufacturer as M ON V.M_ID = M.M_ID
INNER JOIN Part AS P ON R.VIN = P.VIN and R.customerID = P.customerID and
R.start_date = P.start_date and R.username = P.partID
WHERE VIN = '$VIN' AND customerID = '$cusomterID' AND start_date = '$start_date';
```

- If ***Save*** button is clicked:
    - Check if info like Odometer reading and current date has been changed. If so, display the error message.
    - Check if the input is valid such as quantity of parts is a positive number. Check if complete date is after the start date. If not, display the error message.

```
UPDATE Repair
SET complete_date = '$complete_date', labor_charge = '$labor_charge', description =
'$description'
WHERE VIN = '$VIN' and CustomerID = '$CustomerID' and start_date = '$start_date';
```

```
UPDATE Part
SET part_quantity = '$part_quantity', vendor = '$vendor', part_price = '$part_price'
WHERE VIN = '$VIN' and CustomerID = '$CustomerID' and start_date = '$start_date'
and partID = '$partID';
```

- Save the **Repair** Form.

View Sales by color

Abstract Code:

- The login user clicked ***sales by color*** button in **Search Page**.
- Run the **view sales by color** task:
  For each single color inclecluding multiple color that all the vehicles have, find and display the color, number of vehicles sold in past 30 days, number of vehicles sold in past year, and number of vehicles sold over all time. Display by vehicle color ascending.

```
WITH VehicleColorCount (VIN, color, color_cnt) AS
(SELECT VIN, color, count(color) over (partition by VIN) FROM VehicleColor)

,VehicleTrueColor (VIN, true_color) AS
(SELECT DISTINCT VIN,
                CASE WHEN (color_cnt > 1) THEN ('multiple')
                ELSE (color)
                END AS (true_color)
FROM VehicleColorCount)

,SaleColorDays (username, VIN, customerID, color, days_sold) AS
(SELECT username,
                VIN,
                customerID,
                true_color,
                DATEDIFF(day, sold_date, GETDATE()) AS days_sold
FROM Sale INNER JOIN VehicleTrueColor ON Sale.VIN = VehicleTrueColor.VIN)

,SaleByColor(color, 30Days, 1Year, allTime) AS
(SELECT color,
                sum(CASE WHEN days_sold <= 30 THEN 1 ELSE 0 END) AS 30Days,
                sum(CASE WHEN days_sold <= 365 THEN 1 ELSE 0 END) as 1Year,
                count(*) as allTime
FROM SaleColorDays
GROUP BY color)

,AllColors (color) AS
(SELECT DISTINCT color FROM VehicleTrueColor)

SELECT AllColors.color as color,
          ISNULL(30days, 0) as 30Days,
          ISNULL(1Year, 0) as 1Year,
          ISNULL(allTime, 0) as allTime
FROM AllColors LEFT JOIN SaleByColor
ON AllColors.color = SaleByColor.color
ORDER BY AllColors.color ASC;
```

- Click **Close** Button to close the report

View Sales by type

Abstract Code:

- The login user clicked ***sales by type*** button in **Search Page**.
- Run the **view sales by type** task:
  For each vehicle type, find and display the vehicle type, number of vehicles sold in past 30 days, number of vehicles sold in past year, and number of vehicles sold over all time. Display by vehicle type ascending.

```
WITH VehicleWithType(VIN, model_name, model_year, invoice_price, is_sold,
        description, vehicle_type, M_ID, add_date, username) AS
SELECT(V.VIN, model_name, model_year, invocie_price, is_sold, description,
          COALESCE(Car.type, Con.type, Truck.type, VM.type, SUV.type) as
vehicle_type,
          M_ID, add_date, username
FROM Vehicle AS V
    LEFT JOIN Car ON V.VIN = Car.VIN
        LEFT JOIN Convertible AS Con ON V.VIN = Con.VIN
        LEFT JOIN Truck ON V.VIN = Truck.VIN
        LEFT JOIN VanMinivan AS VM ON V.VIN = VM.VIN
        LEFT JOIN SUV ON V.VIN = SUV.VIN
)

,SaleTypeDays (username, VIN, customerID, vehicle_type, sold_days) AS
(SELECT Sale.username,
                Sale.VIN,
                Sale.customerID,
                vehicle_type,
                DATEDIFF(day, sold_date, GETDATE()) AS days_sold
FROM Sale INNER JOIN VehicleWithType AS VWT ON Sale.VIN = VWT.VIN)

,SaleByType (vehicle_type, 30Days, 1Year, allTime) AS
(SELECT vehicle_type,
          sum(CASE WHEN days_sold <= 30 THEN 1 ELSE 0 END) as 30Days,
          sum(CASE WHEN days_sold <= 365 THEN 1 ELSE 0 END) as 1Year,
          count(*) as allTime
FROM SaleTypeDays
GROUP BY vehicle_type)

,AllTypes (vehicle_type) AS
(SELECT DISTINCT vehicle_type FROM VehicleWithType)

SELECT vehicle_type,
          ISNULL(30days, 0) as 30Days,
          ISNULL(1Year, 0) as 1Year,
          ISNULL(allTime, 0) as allTime
```

```
FROM AllTypes LEFT JOIN SaleByType AS SBC
ON AllTypes.vehicle_type = SBC.vehicle_type
ORDER BY vehicle_type ASC;
```

- Click ***Close*** Button to close the report

View Sales by manufacturer

Abstract Code:

- The login user clicked ***sales by manufacturer*** button in **Search Page**.
- Run the **view sales by manufacturer** task:
  For each manufacturer that has any at least a sale over all time, find and display the manufacturer name, number of vehicles sold in past 30 days, number of vehicles sold in past year, and number of vehicles sold over all time. Display by manufacturer name ascending.

```
WITH VehicleManufacturer (VIN, manufacturer_name) AS
(SELECT Vehicle.VIN, name
FROM Vehicle INNER JOIN Manufacturer
ON Vehicle.M_ID = Manufacturer.M_ID)

,SaleManufacturerDays (username, VIN, customerID, manufacturer_name,
days_sold) AS
(SELECT username,
                Sale.VIN,
                customerID,
                name,
                DATEDIFF(day, sold_date, GETDATE()) AS days_sold
FROM Sale INNER JOIN VehicleManufacturer AS VM
ON Sale.VIN = VM.VIN)

SELECT manufacturer_name,
          sum(CASE WHEN days_sold <= 30 THEN 1 ELSE 0 END) as 30Days,
          sum(CASE WHEN days_sold <= 365 THEN 1 ELSE 0 END) as 1Year,
          count(*) as allTime
From SaleManufacturerDays
GROUP BY manufacturer_name
ORDER BY manufacturer_name ASC;
```

- Click ***Close*** Button to close the report

View Gross Customer Income

Abstract Code:

- The login user clicked ***view gross customer income*** button in **Search Page**.
- First, run **view top customers** task:
  Find and display the customer's name (first/last for individuals or business name for business), the date of the first sale or repair start date, the date of their most recent sale or repair start date, their number of sales, their number of repairs, and the gross income of top 15 customers by gross income descending and by last sale/repair start date descending.

```
WITH CustomerRepair (customerID, repair_count, first_repair_date, last_repair_date,
repair_charge) AS
(SELECT R.customerID, count(*),
            min(R.start_date), max(R.start_date),
            sum(labor_charge + part_quantity * part_price)
FROM Repair AS R INNER JOIN Part AS P
          ON R.VIN = N.VIN AND R.customerID = P.customerID AND R.start_date =
P.start_date
GROUP BY R.customerID)

,CustomerSale (customerID, sale_count, first_sale_date, last_sale_date,
sale_charge) AS
(SELECT customerID, count(*), min(sold_date), max(sold_date), sum(sold_price)
FROM Sale
GROUP BY customerID)

,CustomerRepairSale AS
(SELECT
           CASE WHEN (CR.customerID is NOT Null) THEN (CR.customerID)
           ELSE (CS.customerID)
           END AS (customerID)
           first_repair_date,
           first_sale_date,
           last_repair_date,
           last_sale_date,
           repair_count,
           repair_charge,
           sale_count,
           sale_charge
FROM CustomerRepair AS CR FULL OUTER JOIN CustomerSale AS CS
          ON CR.customerID = CS.customerID)

,CustomerExpense AS
(SELECT customerID,
                 min(first_repair_date, first_sale_date) AS first_date,
                 max(last_repair_date, last_sale_date) AS last_date,
                    ISNULL(sale_count, 0) AS sale_count,
                    ISNULL(repair_count, 0) AS repair_count,
                    (ISNULL(repair_charge, 0) + ISNULL(sale_charge, 0)) AS
total_expense
FROM CustomerRepairSale
GROUP BY customerID
ORDER BY ISNULL(repair_charge, 0) + ISNULL(sale_charge, 0) DESC,
max(last_repair_date, last_sale_date) ASC
LIMIT 15)

,CustomerName(customerID, customer_name) AS
(SELECT
            CASE WHEN (driver_license is NOT Null) THEN (I.customerID)
            ELSE (B.customerID)
            END AS (customerID),

      CASE WHEN (driver_license is NOT Null) THEN (CONCAT(first_name,
last_name))
            ELSE (B.name)
            END AS (customer_name)
FROM Individual AS I FULL OUTER JOIN Business AS B ON I.customerID =
B.customerID)

SELECT customer_name, first_date, last_date, sale_count, repair_count,
total_expense
FROM CustomerExpense as CE INNER JOIN CustomerName as CN ON
CE.customerID = CN.customerID
ORDER BY total_expense DESC, last_date DESC;
```

- If a customer's name ('$customer_name') is ***clicked*** in the display:

    - Run **view vehicle sales** task:

      For every sale of a customer that has name '$customer_name',
      Find and display sale date, sold price, VIN, year, manufacturer, model, and salesperson name. The display should be sorted by sale date descending and VIN ascending.

```
WITH CustomerName(customerID, customer_name) AS
(SELECT
            CASE WHEN (driver_license is NOT Null) THEN (I.customerID)
            ELSE (B.customerID)
            END AS (customerID),
      CASE WHEN (driver_license is NOT Null) THEN (CONCAT(first_name,
last_name))
            ELSE (B.name)
            END AS (customer_name)
FROM Individual AS I FULL OUTER JOIN Business AS B ON I.customerID =
B.customerID)

,SelectedCustomerID AS
(SELECT customerID
FROM CustomerName
WHERE customer_name = '$customer_name')

,SaleACustomer(VIN, sold_date, sold_price, saleperson_name) AS
(SELECT Sale.VIN, sold_date, sold_price, CONCAT(first_name, last_name)
FROM Sale
    INNER JOIN Customer ON Sale.customerID = Customer.customerID
    INNER JOIN LoginUser ON Sale.username = LoginUser.username
WHERE Customer.customerID IN SelectedCustomerID)

SELECT sold_date, sold_price, VIN, model_year,
      name AS manufacturer_name, model_name, saleperson_name
FROM  SaleACustomer
     INNER JOIN Vehicle ON SaleACustomer.VIN = Vehicle.VIN
     INNER JOIN Manufacturer ON Vehicle.M_ID = Manufacturer.M_ID
ORDER BY sold_date DESC, Vehicle.VIN ASC;
```

- Run **view vehicle repairs** task:
  For every repair of a customer that has name '$customer_name', find and display start date, end date (end date displays no value if a repair is not finished), VIN, odometer reading, parts cost, labor cost, total cost, and service writer. This listing should be sorted by start date descending, end date descending, and VIN ascending; however, any incomplete repairs should be listed before completed ones with the same sorting criteria.

```
WITH CustomerName(customerID, customer_name) AS
(SELECT
            CASE WHEN (driver_license is NOT Null) THEN (I.customerID)
            ELSE (B.customerID)
            END AS (customerID),

      CASE WHEN (driver_license is NOT Null) THEN (CONCAT(first_name,
last_name))
            ELSE (B.name)
            END AS (customer_name)
FROM Individual AS I FULL OUTER JOIN Business AS B ON I.customerID =
B.customerID)
```

```
,SelectedCustomerID AS
(SELECT customerID FROM CustomerName WHERE customer_name =
'$customer_name'),

,PartsCostACustomer (VIN, customerID, start_date, all_parts_cost) AS
(SELECT VIN, customerID, start_date, sum(part_price * part_quantity)
FROM Part
WHERE customerID IN SelectedCustomerID
GROUP BY VIN, customerID, start_date)

SELECT start_date, end_date, VIN, Odometer, all_parts_cost, labor_cost,
      (all_parts_cost + larbor_cost) as total_cost,
      CONCAT(LU.first_name, ' ', LU.last_name) as servicewriter_name
FROM Repair AS R INNER JOIN PartsCostACustomer AS PCC
          ON R.VIN = PCC.VIN AND R.VIN = PCC.customerID AND R.start_date =
PCC.start_date
          INNER JOIN Vehicle
          ON R.VIN = Vehicle.VIN
          INNER JOIN LoginUser AS LU
          ON Vehicle.username = LU.username
ORDER BY start_date DESC, end_date DESC NULLS FIRST, R.VIN ASC;
```

    - Click ***Close*** Button to close the display

- Click ***Close*** Button to close the report

View Repairs by Manufacturer/Type/Model

Abstract Code:

- The login user clicked ***View Repairs by Manufacturer/Type/Model*** button in **Search Page**.

- Run **View Repairs by Manufacturer** task:
  For every manufacturer, find its repair, calculate total number of repairs, total all parts costs, total all labor costs, total all repairs costs. Display by manufacturer name ascending.

```
WITH PartsCost (VIN, customerID, start_date, all_parts_cost) AS
(SELECT VIN, customerID, start_date, sum(part_price * part_quantity)
FROM Part
GROUP BY VIN, customerID, start_date)

,RepairCost (VIN, customerID, start_date, labor_charge, all_parts_cost, total_cost)
AS
(SELECT R.VIN, R.customerID, R.start_date, labor_charge, all_parts_cost,
            (labor_charge + all_parts_cost) as total_cost
FROM Repair AS R INNER JOIN PartsCost AS PC
ON R.VIN = PC.VIN AND R.customerID = PC.customerID AND R.start_date =
PC.start_date)

,VehicleManufacturer (VIN, manufacturer_name) AS
(SELECT Vehicle.VIN, name
FROM Vehicle INNER JOIN Manufacturer
ON Vehicle.M_ID = Manufacturer.M_ID)

SELECT VM.manufacturer_name AS manufacturer_name,
      count(*) as number_repairs,
      sum(all_parts_cost) as total_parts_cost,
      sum(labor_charge) as total_labor_cost,
      sum(all_parts_cost + labor_charge) as total_cost
FROM RepairCost INNER JOIN VehicleManufacturer as VM ON RepairCost.VIN =
VM.VIN
GROUP BY VM.manufacturer_name
ORDER BY VM.manufacturer_name ASC;
```

- If a manufacturer's name ('$manufacturer_name') is **clicked** in the display:

  - Run **View Repairs by Type/Model** task:

  Find all the repairs of the vehicle that is manufactured by '$manufacturer_name'. For each vehicle type, calculate repair count, parts costs, labor costs, and total costs, and display by total repair count in descending.

Under each vehicle type, for each model, calculate repair counts, parts costs, labor costs, and total costs, and display by total repair counts in descending order.

```
WITH VehicleWithType (VIN, model_name, model_year, invoice_price, is_sold,
         description, vehicle_type, M_ID, add_date, username) AS
SELECT (V.VIN, model_name, model_year, invocie_price, is_sold, description,
           COALESCE(Car.type, Con.type, Truck.type, VM.type, SUV.type) as
vehicle_type,
           M_ID, add_date, username
FROM Vehicle AS V
    LEFT JOIN Car ON V.VIN = Car.VIN
         LEFT JOIN Convertible AS Con ON V.VIN = Con.VIN
         LEFT JOIN Truck ON V.VIN = Truck.VIN
         LEFT JOIN VanMinivan AS VM ON V.VIN = VM.VIN
         LEFT JOIN SUV ON V.VIN = SUV.VIN
)

,RepairAManufacturer (VIN, customerID, start_date, vehicle_type, model_name,
labor_charge) AS
(SELECT VIN, customerID, start_date, vehicle_type, model_name, labor_charge
FROM Repair
         INNER JOIN VehicleWithType AS VWT ON Repair.VIN = VWT.VIN
         INNER JOIN Manufacturer ON Vehicle.M_ID = Manufacturer.M_ID
WHERE Manufacturer.name = '$manufacturer_name')

,PartsCost (VIN, customerID, start_date, parts_charge) AS
(SELECT VIN, customerID, start_date, sum(part_price * part_quantity) as
parts_charge
FROM Part
GROUP BY VIN, customerID, start_date)

,VehicleModelRepair AS
(SELECT vehicle_type, model_name,
                count(*) as model_number_repairs,
                sum(labor_charge) as model_labor_charge,
                sum(parts_charge) as model_parts_charge,
                sum(labor_charge + parts_charge) as model_repair_charge
FROM RepairAManufacturer as RAM INNER JOIN PartsCost AS PC
    ON RAM.VIN = PC.VIN AND RAM.customerID = PC.customerID AND
RAM.start_date = PC.start_date
GROUP BY vehicle_type, model_name)

,VehicleTypeModelRepair AS
(SELECT vehicle_type, model_name,
                model_number_repairs, model_labor_charge, model_parts_charge,
model_repair_charge,
                  sum(model_number_repairs) OVER(PARTITION BY vehicle_type) as
type_number_repairs,
                  sum(model_labor_charge) OVER(PARTITION BY vehicle_type) as
type_labor_charge,
                  sum(model_parts_charge) OVER(PARTITION BY vehicle_type) as
type_parts_charge,
                  sum(model_repair_charge) OVER (PARTITION by vehicle_type) as
type_repair_charge
FROM VehicleModelRepair)

SELECT vehicle_type, model_name,
            model_number_repairs, model_labor_charge, model_parts_charge,
model_repair_charge,
            type_number_repairs, type_labor_charge, type_parts_charge,
type_repair_charge
FROM VehicleTypeModelRepair
ORDER BY type_number_repairs DESC, vehicle_type DESC, model_number_repairs
DESC;
```

    - Click ***Close*** Button to close the display

- Click ***Close*** Button to close the report

View Below Cost Sales

Abstract Code:

- The login user clicked **View Below Cost Sales** button in **Search Page**.
- Run the **View Below Cost Sales** task:
  Find the sales for which the vehicle has a sold price lower than invoice price.
  For each sale,
  find the sale date, invoice price, sold price, name, salesperson, and calculate sold/invoice price ratio (%).
  Sort by sale date descending and ratio descending.
  Display each sale.

```
WITH CustomerName(customerID, customer_name) AS
(SELECT
          CASE WHEN (driver_license is NOT Null) THEN (I.customerID)
          ELSE (B.customerID)
          END AS (customerID),

     CASE WHEN (driver_license is NOT Null) THEN (CONCAT(first_name,
last_name))
          ELSE (B.name)
          END AS (customer_name)
FROM Individual AS I FULL OUTER JOIN Business AS B ON I.customerID =
B.customerID)

SELECT sold_date, invoice_price, sold_price,
     CONCAT(LU.first_name, ' ', LU.last_name) as saleperson_name,
     CN.customer_name as customer_name,
     sold_price/invoice_price * 100 AS ratio
FROM Sale
        INNER JOIN Vehicle ON Sale.VIN = Vehicle.VIN
        INNER JOIN LoginUser AS LU ON Sale.username = LU.username
        INNER JOIN CustomerName AS CN ON Sale.customerID = CN.customerID
WHERE sold_price < invoice_price
ORDER BY sold_date DESC, sold_price/invoice_price DESC;
```

- Click **Close** Button to close the report

View Average Time In Inventory

Abstract Code:

- The login user clicked **View Average Time In Inventory** button in **Search Page**.
- Run **View Average Time In Inventory** task:

```
For each vehicle type in ascending order:
        Find all sales that are related to the vehicle type.
        If no sale is found,
                Display vehicle type, and N/A.
        Else:
                Calculate the average amount of time a vehicle remains in
        inventory.
                Display vehicle type, and average amount of time in days.
```

```
WITH VehicleWithType (VIN, model_name, model_year, invoice_price, is_sold,
        description, vehicle_type, M_ID, add_date, username) AS
SELECT (V.VIN, model_name, model_year, invocie_price, is_sold, description,
          COALESCE(Car.type, Con.type, Truck.type, VM.type, SUV.type) as
vehicle_type,
          M_ID, add_date, username
FROM Vehicle AS V
    LEFT JOIN Car ON V.VIN = Car.VIN
         LEFT JOIN Convertible AS Con ON V.VIN = Con.VIN
         LEFT JOIN Truck ON V.VIN = Truck.VIN
         LEFT JOIN VanMinivan AS VM ON V.VIN = VM.VIN
         LEFT JOIN SUV ON V.VIN = SUV.VIN
)

,DaysInInventory (VIN, vehicle_type, days_in_inventroy) AS
(SELECT Sale.VIN,
                vehicle_type,
                DATEDIFF(day, sold_date, GETDATE()) + 1
FROM Sale INNER JOIN VehicleWithType AS VWT ON Sale.VIN = VWT.VIN)

,AvgDaysInInventory (vehicle_type, avg_days_in_inventroy) AS
(SELECT vehicle_type, CAST(avg(days_in_inventroy) AS varchar)
FROM DaysInInventory
GROUP BY vehicle_type)

,AllTypes (vechile_type) AS
(SELECT DISTINCT vehicle_type FROM VehicleWithType)

SELECT type, ISNULL(avg_days_in_inventroy, 'NA') as avg_days_in_inventroy
FROM AllTypes LEFT JOIN AvgDaysInInventory AS ADII
ON AllTypes.vechile_type = ADII.vechile_type
ORDER BY AllTypes.vechile_type ASC;
```

- Click **Close** Button to close the report

View Parts Statistics

Abstract Code:

- The login user clicked **View Parts Statistics** button in **Search Page**.
- Run **View Parts Statistics** task:
  Find all parts in all repairs.
  For each vendor,
      count number of parts and calculate total dollar amount spent.
  Display vendor name, the number of parts, and the total dollar amount spent in descending.

```
SELECT vendor_name,
      sum(part_quantity) as number_parts,
      sum(part_price * part_quantity) as total_parts_expense
FROM Part
GROUP BY vendor_name
ORDER BY sum(part_price * part_quantity);
```

- Click **Close** Button to close the report.

View Monthly Sales

Abstract Code:

- The login user clicked **View Monthly Sales** button in **Search Page**.
- Run **View sales over time** task:

```
Find all sales.
For each month,
    count number of vehicles sold for each month.
    calculate total sales income,
    calculate total net income (invoice price – sold price).
    calculate sold price/invoice ratio as percentage.
Order by year and month descending.
Display this information.
```

```
WITH SaleByMonth (VIN, year, month, sold_price, invoice_price) AS
(SELECT Sale.VIN,
                YEAR(sold_date) AS year,
                Month(sold_date) AS month,
                sold_price,
                invoice_price
FROM Sale INNER JOIN Vehicle ON Sale.VIN = Vehicle.VIN)

SELECT year,
      month,
      count(*) as number_sale,
      sum(sold_price) as total_sale_income,
      sum(sold_price - invoice_price) as total_net_income,
      (sum(sold_price) / sum(invoice_price)) as ratio
FROM SaleByMonth
GROUP BY year, month
ORDER BY year DESC, month DESC;
```

- If a year/month ('$year', and '$month') is **clicked** in the display:
  Run **view salesperson rank** task:

```
Select sales whose year or month matches the user's selection.
For every salespeople,
    count number of vehicles sold, and total sales.
Sort by total vehicles sold descending followed by total sales descending.
Display salesperson's first name, last name, number of vehicles sold, and
total sales.
```

```
WITH SaleByMonth (saleperson_name, sold_price) AS
(SELECT CONCAT(LU.first_name, ' ', LU.last_name) as saleperson_name,
sold_price
FROM Sale INNER JOIN LoginUser AS LU ON Sale.username = LU.username
WHERE YEAR(sold_date) = '$year' AND MONTH(sold_date) = '$month')
```

```
SELECT saleperson_name,
            count(*) as total_vehicles_sold,
            sum(sold_price) as total_sales
FROM SaleByMonth
GROUP BY saleperson_name
ORDER BY count(*) DESC, sum(sold_price) DESC;
```

    - Click **Close** Button to close the display.
- Click **Close** Button to close the report.