Phase 2 Report

CS 6400-Fall 2021

Team 081

Table of Contents:

Abstract Code Legend **3**

Abstract Code **4**

- Main Landing 4
- LOGIN 5
- Get Number Of Total Vehicles Available 6
- Search Vehicle by VIN 7
- Search Vehicles 8
- Filter By Sold/Unsold/All Vehicles 13
- Add Vehicle 15
- View Vehicle 17
- Lookup Customer 23
- Add Customer 24
- Enter Sale 25
- Create Repair 26
- Add Part 27
- Complete Repair 28
- Update Labor Charge 29
- View Repair 30
- View/Generate Reports 32
- Sales By Color: 32
- Sales By Type: 33
- Sales By Manufacturer: 34
- Gross Customer Income: 35
- Average Time In Inventory: 38
- Part Statistics: 39
- Below Cost Sales: 40
- Repairs by Manufacturer/Type/Model 41
- Monthly Sales 43

Abstract Code Legend

Form - bold, underline

Task - bold

Button - bold, italics

Attribute - italics

TableName - Blue and First letter Uppercase

$variableNames - camelcase, preceded by “$” and green.

Abstract Code

- **Relational DBMS Used = MS SQL**

Main Landing

Abstract Code

- User lands on the **Main Landing** form.
- Run ***Number of Total Vehicles Available*** task and show the value on the page.

```
SELECT count(Vehicle.VIN) AS number_of_tot_vehicles
FROM Vehicle
LEFT OUTER JOIN Sale
  ON Vehicle.VIN=Sale.VIN
WHERE Sale.Sale_price IS NULL;
```

- Show search fields for *Vehicle_type* , *Manufacturer_name*, *Model_year* , *Color* **Dropdowns,** ***Search Vehicles*** button**,** ***List_price*** search tab**,** ***Keyword*** search tab , and ***Log In*** link.

- When the user logs in,in addition to the above , on the **Main Landing** form
  - If the logged in user is **MANAGER**,Upon:
    - Click ***View/Generate reports*** link- Jump to **View/Generate** task
    - Click ***sold/unsold/all vehicles*** dropdown- Jump to **Filter By Sold/Unsold/All Vehicles** task
    - Click ***Search Vehicle by VIN*** search tab- Jump to **Search Vehicle by VIN** task.
  - if the logged in user is **INVENTORY CLERK**,Upon:
    - Click ***Add vehicle*** button/link- Jump to **Add Vehicle** form,
    - Click ***Search Vehicle by VIN*** search tab-Jump to **Search Vehicle by VIN** task
  - if the logged in user is a **SALESPERSON**, Upon:
    - Click ***Search Vehicle by VIN*** search tab-Jump to **Search Vehicle by VIN** task.
  - if the logged in user is a **SERVICE WRITER**, Upon:
    - Click ***Search Vehicle by VIN*** search tab-Jump to **Search Vehicle by VIN** task.
    - Click ***Repair*** link/button -Jump to **Repair** form.
  - if the logged in user is the **OWNER**, run all tasks as above and show all the tabs and links viewable to each individual logged in users.

LOGIN

- The user enters $username and $password in the corresponding fields
- If data validation is correct for both fields user and password then:
    - When the ***Login*** button is clicked:

```
SELECT  password FROM EmployeeUser WHERE username = $username;
```

    - If $username record is found but $password does not match
        - Go to **Login** form showing an error message
    - If $username and $password are correct
        - Get the user role and persist information along with the session
        - Check role of the current user and go to **Main Landing** form showing fields according to the role
- Else if both user and password are incorrect, display the **Login** form with a message indicating the error.

Get Number Of Total Vehicles Available

Abstract Code

- When a user lands on the **Main Landing** form, run the **Get Number Of Total Vehicles Available** task**.**
- **Get Number Of Total Vehicles Available-** Jumps to the **Get Total Vehicles Available** task**.**
    - Run a query on **SALE** table and **VEHICLE** table select count of Vehicle. (*) using *VIN* as a join where the Sale_price is null.
    - Display count.

```
SELECT count(Vehicle.VIN) AS [Number Of Total Vehicles ]
FROM Vehicle LEFT OUTER JOIN Sale  ON Vehicle.VIN=Sale.VIN
WHERE Sale.Sale_price IS NULL;
```

Get Options for Search Dropdowns

Abstract Code

- When a user lands on the **Main Landing** form, run the **Get Options for Search Dropdowns** task**.**

- **Main Landing**.

//Populate *Vehicle type* , *Manufacturer*, *Year, Color* ***dropdowns*****.**

```
SELECT Manufacturer_name FROM Manufacturer;

--DropDown for Vehicle Type
SELECT DISTINCT  Vehicle_type
FROM (
                        SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION    SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION    SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION    SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION    SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType

SELECT DISTINCT (Year) from Vehicle;
SELECT DISTINCT (Color) from Color;
```

- if no button is clicked, do nothing. Otherwise, users may select different options from the dropdowns and may enter the list price and/or keyword.
    - User hits ***Search Vehicles*** button,
        - Jump to the **Search Vehicles** task.

Search Vehicle by VIN

Abstract Code

- When the ***Search Vehicle by VIN*** search tab is clicked:
  - User clicks on the ***Search Vehicle by VIN*** and enters *VIN*.
  - If data validation is successful, then:
    - When enter button is clicked:

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name,V.Description
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price
FROM Vehicle v
LEFT JOIN (
                    SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION  SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION  SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION  SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION  SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
AND VIN='$VIN'
ORDER BY V.VIN ASC;
```

      - If a record is found, go to the **Vehicle Details** form.
      - Else:
        - Go back to **Main Landing** form, with the error message "Sorry, no such Vehicle found!".
  - Else:
    - Display message "Invalid VIN".

Search Vehicles

Abstract Code

- When the ***Search Vehicles*** button is clicked-Jump to **Search Vehicles** task.
- Run the **Search Vehicles** task: query for information on the **VEHICLE, SALE** using *VIN* as join.
- select *VIN*, *Year*, *Model_name*, *Manufacturer_name*, *Colors*, *List_price*, *Description*, *List_price* from **VEHICLE** corresponding to chosen criteria by user. Additionally the type of the vehicle will be used for filtering.

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name,V.Description
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price
FROM Vehicle v
LEFT JOIN (
                    SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION  SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION  SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION  SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION  SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
ORDER BY V.VIN ASC;
```

  - if vehicle has single color:
    - In ascending order with respect to *VIN*, display all vehicles that match the selected criteria from ***dropdowns*** (the *Color*, *Vehicle_type*, *Year*, *Model_name*, *Manufacturer_name*, List_price, keyword and *VIN*) If a keyword was entered and matched the description, indicate this with a checkbox.If the user selects one individual result, jump to **Vehicle Details** form.
  - Else if a vehicle has multiple colors:
    - In ascending order with respect to *VIN*, display all Vehicles that match the selected criteria entered (the *Color*, Vehicle_type,*Year*,*Model_name* , *Manufacturer_name*, *List_price*, keyword and *VIN*) and a single row with all colors listed.If a keyword was entered and matched the description, indicate this with a checkbox. If the user selects one individual result, jump to **Vehicle Details** form.
  - Else:
    - If no record is found matching the selected criteria ,go back to **Main Landing** form and display an error message: “Sorry, it looks like we don’t have that in stock!”

- Searching can be done on the following criteria

Vehicle type : user will select an option from the **dropdown** the value will be stored in the variable $vehicleType

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name,V.Description
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price
FROM Vehicle v
LEFT JOIN (
                    SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION  SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION  SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION  SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION  SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
AND VehicleType.Vehicle_type = '$VehicleType'
ORDER BY V.VIN ASC;
```

• Manufacturer: user will select an option from the **dropdown** the value will be stored in the variable $Manufacturer

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name,V.Description
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price
FROM Vehicle v
LEFT JOIN (
                    SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION  SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION  SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION  SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION  SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
AND v.Manufacturer_name = '$Manufacturer'
ORDER BY V.VIN ASC;
```

• Model year: user will select an option from the **dropdown** the value will be stored in the variable $Model_Year

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name,V.Description
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price
FROM Vehicle v
LEFT JOIN (
                        SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION  SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION  SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION  SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION  SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
AND v.Year = '$Model_Year'
ORDER BY V.VIN ASC;
```

• Color: user will select an option from the **dropdown** the value will be stored in the variable $Color

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name,V.Description
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price
FROM Vehicle v
LEFT JOIN (
                        SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION  SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION  SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION  SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION  SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
AND  ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN) like '%$Color%'
ORDER BY V.VIN ASC;
```

- List price (either greater than and/or less than an entered value)
    - Greater Than

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name,V.Description
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price
FROM Vehicle v
LEFT JOIN (
                    SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION  SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION  SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION  SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION  SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
AND v.List_price > '$List_Price'
ORDER BY V.VIN ASC;
```

- user search List Price less than

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name,V.Description
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price
FROM Vehicle v
LEFT JOIN (
                    SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION  SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION  SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION  SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION  SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s) AND
v.List_price < '$List_Price'
ORDER BY V.VIN ASC,
```

- user search List Price by Entered Value

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name,V.Description
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price
FROM Vehicle v
LEFT JOIN (
                    SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION  SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION  SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION  SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION  SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s) AND
v.List_price = '$List_Price'
ORDER BY V.VIN ASC;
```

• User enters a keyword and it's stored in the variable $keyword, the system will search on specific columns (either entirely or as a substring).

Keyword, which searches the manufacturer, model year, model name and description fields.

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name,V.Description
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price
FROM Vehicle v
LEFT JOIN (
                    SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION  SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION  SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION  SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION  SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
AND (v.Manufacturer_name like '%$keyword%'
OR v.Year like '%$keyword%'
OR v.Model_name like '%$keyword%'
OR v.Description like '%$keyword%')
ORDER BY V.VIN ASC;
```

Filter By Sold/Unsold/All Vehicles

Abstract Code

- When user clicks on ***Filter by sold/unsold/all***
  - Populate ***Filter by sold/unsold/all*** dropdown. A dropdown opens, user selects either sold or unsold or All vehicles.
  - if the user clicks on **unsold**:
    - Run a query on the **SALE**, **VEHICLE** tables, to find Unsold Vehicles

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name,V.Description
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION  SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION  SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION  SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION  SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
ORDER BY V.VIN ASC;
```

  - if the user clicks on **sold**. (for Managers) Run a query on the **SALE**,**VEHICLE** tables.Use VIN as a join

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name,V.Description
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION  SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION  SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION  SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION  SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
WHERE v.VIN IN( SELECT s.VIN FROM Sale s)
ORDER BY V.VIN ASC;
```

- User select **all** vehicles

    - Run a query on **VEHICLE,MANUFACTURER** Table and display all Vehicles, if **all vehicles** selected from dropdown.

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name,V.Description
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION  SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION  SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION  SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION  SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
ORDER BY V.VIN ASC;
```

Add Vehicle

Abstract Code

- Show **Main Landing** form**.**
- Upon:
    - Click ***Add Vehicle*** button **-** Jump to the **New Vehicle** form**.**
- **INVENTORY CLERK** inputs all prompts for a complete new vehicle description**.**
- **INVENTORY CLERK** name is also recorded
- Upon:
    - Click ***Submit New Vehicle*** button - perform the following:
        - check that all fields are complete
            - if input fields are empty, return “missing field” alert, show which fields are missing by outlining them in red, and allow the clerk to fill in and submit again.
        - for all inputs, check that types match expected schema
            - If schema doesn’t match, return “wrong schema” alert, outline which fields have wrong schema. Allow the clerk to correct and submit again.
        - if all checks pass, submit the record to the database and perform **Add Vehicle** task

```
INSERT INTO Vehicle
(VIN,Year,Model_name,Description,Invoice_price,List_price,Inventory_date
,Manufacturer_name,Username)
VALUES
('$VIN', '$year', '$model_name', '$description', '$invoice_price', '$list_price', GETDATE(),
'$manufacturer_name', '$username')
```

Insert Vehicle type in the corresponding table.

- if Inventory clerk select Car insert Attributes for car:

```
INSERT INTO Car(VIN,Doors_count) VALUES ($VIN,$Doors_count)
```

- if Inventory clerk select Truck insert Attributes for Truck:

```
INSERT INTO Truck(VIN,Cargo_capacity,Cargo_cover_type,Axle_count) VALUES
('$VIN','$Cargo_capacity','$Cargo_cover_type','$Axle_count')
```

- if Inventory clerk select Convertible insert Attributes for Convertible:

```
INSERT INTO Convertible (VIN,Roof_type,Back_seat_count) VALUES
('$VIN','$Roof_type','$Back_seat_count')
```

- if Inventory clerk select SUV insert Attributes for SUV

```
INSERT INTO SUV(VIN,Drivetrain_type,Cupholder_count) VALUES
('$VIN','$Drivetrain_type','$Cupholder_count')
```

- if Inventory clerk select VanMinivan insert Attributes for VanMinivan:

```
INSERT INTO VanMinivan(VIN,Has_driver_back_door) VALUES ('$VIN','$Has_driver_back_door')
```

View Vehicle

Abstract Code

- Show **Main Landing** form
- Upon:
  - Click **Vehicle** link - return **View Vehicle** form
- Perform **View Vehicle** task and run the following steps/query to populate the form:
  - Check user role
  - If the role is anonymous, sales person, service writer:
    - select *VIN*, *Vehicle_type*, *Year*, *Model_name*, *Manufacturer_name*, *Color*, *List_price*, *Description* from

If the user select a Vehicle type Car from the main search display details of the vehicle

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price,V.Description
,c.Doors_count
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION    SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION    SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION    SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION    SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
LEFT JOIN Car c ON v.VIN=C.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
AND v.VIN='$VIN';
```

If the user select a Vehicle type truck display detail and the corresponding vehicle Attributes

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price,V.Description
,t.Cargo_capacity,t.Cargo_cover_type,t.Axle_count
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION    SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION    SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION    SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION    SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
LEFT JOIN Truck t ON v.VIN=t.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
AND v.VIN='$VIN';
```

If the user select a Vehicle type SUV display the details and the corresponding vehicle Attributes

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price,V.Description
,t.Cupholder_count,t.Drivetrain_type
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION    SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION    SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION    SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION    SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
LEFT JOIN SUV t ON v.VIN=t.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
AND v.VIN='$VIN';
```

If the user select a Vehicle type Convertible display the details and the corresponding vehicle Attributes

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price,V.Description
,t.Back_seat_count,t.Roof_type
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION    SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION    SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION    SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION    SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
LEFT JOIN Convertible t ON v.VIN=t.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
AND v.VIN='$VIN';
```

If the user select a Vehicle type Convertible display the details and the corresponding vehicle Attributes

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price,V.Description
,t.Has_driver_back_door
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION    SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION    SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION    SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION    SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
LEFT JOIN VanMinivan t ON v.VIN=t.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
AND v.VIN='$VIN';
```

- If the role is Inventory Clerk :
  - select *VIN*, *Vehicle_type*, *Year*, *Model_name*, *Manufacturer_name*, *Color*, *List_price*, *Description, Invoice_price* from **VEHICLE**

If the user select a Vehicle type Car display the details and the corresponding vehicle Attributes

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price,V.Description,v.Invoice_price
,c.Doors_count
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION     SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION     SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION     SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION     SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
LEFT JOIN Car c ON v.VIN=C.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
AND v.VIN='$VIN';
```

If the user select a Vehicle type Truck display the details and the corresponding vehicle Attributes

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price,V.Description,v.Invoice_price
,t.Cargo_capacity,t.Cargo_cover_type,t.Axle_count
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION     SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION     SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION     SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION     SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
LEFT JOIN Truck t ON v.VIN=t.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
AND v.VIN='$VIN';
```

If the user select a Vehicle type SUV display the details and the corresponding vehicle Attributes

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN)
,v.List_price,V.Description,v.Invoice_price
,t.Cupholder_count,t.Drivetrain_type
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION     SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION     SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION     SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION     SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
LEFT JOIN SUV t ON v.VIN=t.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
AND v.VIN='$VIN';
```

If the user select a Vehicle type Convertible display the details and the corresponding vehicle Attributes

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN),v.List_price,V.Description,v.Invoice_price
,t.Back_seat_count,t.Roof_type
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION     SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION     SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION     SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION     SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
LEFT JOIN Convertible t ON v.VIN=t.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
AND v.VIN='$VIN';
```

If the user select a Vehicle type VanMinivan display the details and the corresponding vehicle Attributes

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Manufacturer_name,v.Model_name
, color = ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN),v.List_price,V.Description,v.Invoice_price
,t.Has_driver_back_door
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION     SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION     SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION     SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION     SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
LEFT JOIN VanMinivan t ON v.VIN=t.VIN
WHERE v.VIN NOT IN( SELECT s.VIN FROM Sale s)
AND v.VIN='$VIN';
```

- If the role is Managers or Roland Around:
  - select *VIN*, *Vehicle_type*, *Year*, *Model_name*, *Manufacturer_name*, *Color*, *List_price*, *Description*, *Invoice_price*, *Inventory_date*, from **VEHICLE**.
  - If the vehicle has been sold:
    - select *Sale_date*, *Sale_price* from **SALE**
    - select all but *Drivers_lices_nr* and *TIN* from **CUSTOMER** where *Drivers_lices_nr* /*TIN* matches each customer for each vehicle sold.

```
SELECT v.VIN
,VehicleType.Vehicle_type
,v.Year,v.Model_name,v.Manufacturer_name
, ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=v.VIN) AS COLOR
,v.List_price, v.Description,Inventory_date,(ic.First_name + ' ' + ic.Last_name ) AS
Inventory_Clerk,v.Invoice_price
,s.Sale_date ,s.Sale_price , (e.First_name + ' ' + e.Last_name) as SalesPersonName
,cp.CustomerName as BuyerName
,c.Phone_number as BuyerPhone,c.Email as BuyerEmail,c.Street_address as
BuyerAdress,c.State,c.Postal_code
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION  SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION  SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION  SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION  SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
LEFT JOIN Sale s ON v.VIN=s.VIN
LEFT JOIN EmployeeUser e ON s.Username=e.Username
LEFT JOIN EmployeeUser ic ON v.Username=ic.Username
LEFT JOIN Customer c ON s.Customer_id=c.Customer_id
LEFT JOIN (SELECT p.Customer_id, (p.First_name + ' ' + p.Last_name)as CustomerName FROM Person p
                        UNION
                        SELECT b.Customer_id, b.Business_name as CustomerName FROM Business b) as
CP
        ON CP.customer_id=c.customer_id
WHERE v.VIN='$VIN'
```

- If the vehicle has repairs (**Repairs Section**)
  - select the customer *Name* (*First_name* and *Last_name* for individuals, or *Buisness_name* for companies), the *Service_writer* who entered the repair, and the repair's *Start_date*, *Completion_date*, *Labor_charges*, and *Total_cost*.

```
SELECT
cp.CustomerName
,(e.First_name + ' ' + e.Last_name) as ServiceWriter
,r.Start_date, r.Completion_date, r.Labor_charges, r.Total_cost
FROM Repair r
LEFT JOIN EmployeeUser e ON r.Username=e.Username


LEFT JOIN Customer c ON r.Customer_id=c.Customer_id
LEFT JOIN (SELECT p.Customer_id, (p.First_name + ' ' + p.Last_name)as CustomerName FROM Person p
                                UNION
                                SELECT b.Customer_id, b.Business_name as CustomerName FROM Business b) as CP
        ON CP.customer_id=c.customer_id
WHERE r.VIN='$VIN'
```

Lookup Customer

Abstract Code

- User enters *drivers_license or tax TIN* in the input field.
- If data validation is correct, then:
- Click ***Enter*** button:
    - If *drivers_license* is found

```
SELECT  First_name, Last_name, Phone_number, Email, Street_address, City, State,
Postal_code FROM Customer INNER JOIN Person
ON Customer.customer_id = Person.customer_id
```

    - If *TIN* is found

```
SELECT  Contact_name, Contact_title, Business_name, Phone_number, Email, Street_address, City,
State, Postal_code FROM Customer INNER JOIN Business
ON Customer.customer_id = Business.customer_id
```

    Else
    Display appropriate error message with ***Add Customer*** button

- Else input field is invalid, display input field again with message “Try again”.

Add Customer

Abstract Code

- User clicks ***Add Customer*** button.
- User enters *phone* ($phone)*, address* ($address) and *email* ($email) in the input field.

```
INSERT INTO Customer (phone_number, email, street_address, city, state, postal_code)
VALUES ( '$phone_numer', '$email', '$street_address', '$city', '$state', '$postal_code')
```

- If **CUSTOMER** is Person, then
    - User clicks on ***Individual*** button

```
INSERT INTO  Person (driver_license, customer_id, first_name, last_name) VALUES
('$driver_license','$customer_id','$first_name', '$last_name');
```

- Else
    - User clicks on ***Business*** button

```
INSERT INTO Business(TIN, customer_id, contact_name, contact_title, Business_name)
VALUES ('$TIN', '$customer_id', '$contact_name', '$contact_title', '$Business_name');
```

- User click ***Add*** button, the above query adds the new customer.

Enter Sale

Abstract Code

- User bring vehicle VIN information from **Search Vehicle** Form
- User lookup for Customer using search
    - Run the **Lookup Customer** task using *driver's license or tax ID*
        - If customer exists
            - Show customer information
        - If the customer does not exist
            - Add customer using **Add Customer** task
- The user enters the *Sale Price* value
- The user enters *Sale Date*
- When the User clicks ***Enter Sale*** button
    - Get $userName from the information of the current user using the system
    - Check role of the current user
    - Price Sale is validated
        - If $priceSale is not present or is not numeric
            - display error
        - If the role of the current user is Sales Person then
            - If the $soldprice is less than or equal to 95% of the Invoice price
                - Show error message
        - Else if the role is Owner then
            - Allow inserting any numeric value
    - The sale Date is validated
        - If $saleDate date is not present
            - display error
        - If $saleDate is greater than the current date
            - display error
    - Insert VIN, Username, Sale Price, Sale Date, Username into **SALE** Table.

```
INSERT INTO Sale (VIN,Username,Customer_id,Sale_price,Sale_date)
VALUES('$VIN','$Username','$Customer_id','$Sale_price','$Sale_date')
```

- Jump to **Main Landing** form

Create Repair

Abstract Code

- User enters the *Description* and *Odometer_reading* in the input fields
- User clicks the ***Create Repair*** button
- *Odometer_reading* is validated to be numerical
    - If *Odometer_reading* is not numerical
        - Display error message
- *$Start_date* set to the current date
- *VIN*, *Description*, *Start_date*, and *Odometer_reading* inserted into the **REPAIR** table
    - Insert *VIN, Start_date,* and *DLN* or *TIN* into the appropriate table

```
INSERT INTO Repair (VIN,Customer_id,Start_date,Description, Odometer_reading,Username)
VALUES('$VIN','$Customer_id','$Start_date','$Description','$Odometer_reading','$Username');
```

Add Part

Abstract Code

- User enters the *Part_name*, *Vendor_name*, *Price,* and *Quantity* in the input fields
- User clicks the ***Add Part*** button
- *Price* is validated to be numerical
    - If *Price* is not numerical
        - Display error message
- *Quantity* is validated to be an integer
    - If *Quantity* is not an integer
        - Display error message
- *Part_name* is validated to be present
    - If *Part_name* is not present
        - Display error message
- *Vender_name* is validated to be present
    - If *Vender_name* is not present
        - Display error message
- *VIN*, *Start_date*, *Part_name, Vendor_name*, *Price*, and *Quantity* inserted into the **PART** table

```
INSERT INTO Part(VIN ,Customer_id ,Start_date ,Part_number ,Vendor_name ,Quantity  ,Price)
VALUES('$VIN' ,'$Customer_id' ,'$Start_date' ,'$Part_number' ,'$Vendor_name' ,'$Quantity' ,'$Price' );
```

- Jump to **Show Repair** task

Complete Repair

Abstract Code:

- User selects the repair to mark as complete this will return the values of the repair to be updated
- User clicks the ***Complete Repair*** button
- $completionDate set to the current date
- *Completion_date* is updated to $completionDate on the **REPAIR** table
- USING the VIN start_date and customer_id information, update values

```
UPDATE Repair SET Labor_charges = '$labor_charges' ,Total_cost = '$total_cost',
Description = '$description' ,Completion_date = '$completion_date' ,Username = '$username'
WHERE VIN='$VIN' AND start_date='$start_date' AND Customer_id='$customer_id'
```

- Jump to **Main Landing** form

Update Labor Charge

Abstract Code

- User enters the *Labor_charges* in the input fields
- User clicks the **Update *Labor Charge*** button. This will also return the current labor charge in the Repair table. We perform a query and save results to $previousLaborCharges variable.

```
SELECT Labor_charges
FROM Repair
WHERE VIN = '$currentVin' AND Start_date = '$currentRepairStartDate'
```

- *Labor_charges* is validated to be numerical
    - If *Labor_charges* is not numerical
        - Display error message
- If user is not owner
    - *Labor_charges* is validated to greater than the current value
        - If *Labor_charges* < $previousLaborCharges
            - Display error message
- For the current repair, save the following variables to use in the query: $newLaborCharges, $currentVin, $currentRepairStartDate
- Since the requirements document mentions “Any updates to labor charges cannot be less than their previous value.” We assume that when labor charges are entered, it’s the updated total labor charge (old+new).
- *Labor_charges* is updated on the **REPAIR** table

```
UPDATE Repair
SET Labor_charges = '$newLaborCharges'
WHERE VIN = $currentVin AND Start_date = '$currentRepairStartDate'
```

View Repair

Abstract Code

- Select *Labor_charges*, *Description*, *Odometer_reading*, *Start_date* from **REPAIR** table where the *VIN* matches the $currentVin entered, repair start date matches $currentRepairStartDate and *Completion_date* is NOT null

```
SELECT Start_date, Completion_date, Customer_id, Username, Description, Odometer_reading
Labor_charges, Total_cost, (Total_cost-Labor_charges) AS Part_cost
FROM Repair
WHERE VIN = '$currentVin' AND
        Completion_date IS NOT NULL
```

- If no results are returned
    - Display **CUSTOMER** input panel
    - If **CUSTOMER** is added or selected
        - Display *Description*, and *Odometer_reading* input panels and ***Create Repair*** button, jump to **Create Repair** task.
- Else if results are returned
    - Get the **CUSTOMER** for the **REPAIR** using the *Start_date* and *VIN* number of the **REPAIR** returned
        - If **CUSTOMER** is an **INDIVIDUAL PERSON**
            - Get customer *First_name* and *Last_name*

```
SELECT  First_name, Last_name,
FROM Individual Person
WHERE Customer_id = '$Customer_id'
```

            - If result is >0,
            - Display *First_name* and *Last_name*

        - If **CUSTOMER** was not found in IndividualPerson table (result==0), search in the Business Table**.**
            - Get customer *Business_name*
            - Display *Business_name*

```
SELECT  Business_name
FROM Business
WHERE Customer_id = '$Customer_id'
```

- Get the *Quantity*, *Vendor_name*, *Part_name*, and *Price* for the list of **PART** that match the *Start_date* and *VIN* number of the **REPAIR** returned
  - For each **PART**
    - Display *Quantity*, *Vendor_name*, *Part_number*, and *Price*

```
SELECT Quantity, Vendor_name, Part_number, Price
FROM Repair
WHERE VIN = '$currentVin' AND
        Start_date = '$currentRepairStartDate'
```

- If vehicle repair is not finished, perform the following:

  Display *Description*, *Odometer_reading*, and *Start_date*

```
SELECT Display Description, Odometer_reading, and Start_date
WHERE VIN = '$currentVin' AND
        Start_date = '$currentRepairStartDate' AND
        Completion_date IS NULL
```

- Allow user to add parts:

  Display **PART** input dialogs and ***Add Part*** button, jump to **Add Part** task.

- Display *Labor_charges* in labor charge input and enable editing. Click on ***Update Labor Charges*** button, jump to the **Update Labor Charge** task.
- To complete a repair, click ***Complete Repair*** buttons, jump to **Complete Repair Task**.

View/Generate Reports

Sales By Color:

Abstract Code

When selecting a certain report, a drop down menu opens, select the required report type.
Upon (all reposts follow):

- Clicking **Sales by Color**- Jump to **Sales by Color** task.
    - Perform Query of the **SALE** and **VEHICLE** records:
        - Filter only vehicles that have been sold.
        - Count rows in each group/color where *Sale_date* > current date -30 (for the last month's sales).
            - $targetDate = $currentDate - 30days
            - $newColName = Count*_previous_30_days*
        - Count rows in each group/color where *Sale_date* > current date -365 (for the last year's sales)
            - $targetDate = $currentDate -365days
            - $newColName = Count*_previous_year*

get the last available sale date into $MaxSaleDate

```
SELECT MAX(Sale_date) from Sale
```

```
SELECT CarColor
,count(CASE WHEN s.Sale_date > $MaxSaleDate-30 THEN s.Sale_date ELSE NULL END) LastMonth
,count(CASE WHEN s.Sale_date > $MaxSaleDate-365 THEN s.Sale_date ELSE NULL END) LastYear
,count(s.Sale_date) Alltime
FROM (
SELECT    'Aluminum' AS       CarColor
UNION    SELECT    'Beige' AS            CarColor UNION    SELECT    'Black' AS            CarColor
UNION    SELECT    'Blue'AS   CarColor UNION    SELECT    'Brown' AS            CarColor
UNION    SELECT    'Bronze' AS           CarColor UNION    SELECT    'Claret' AS           CarColor
UNION    SELECT    'Copper' AS           CarColor UNION    SELECT    'Cream' AS            CarColor
UNION    SELECT    'Gold' AS  CarColor UNION    SELECT    'Gray' AS  CarColor
UNION    SELECT    'Green' AS            CarColor UNION    SELECT    'Maroon' AS           CarColor
UNION    SELECT    'Metallic' AS         CarColor UNION    SELECT    'Navy' AS CarColor
UNION    SELECT    'Orange' AS           CarColor UNION    SELECT    'Pink' AS  CarColor
UNION    SELECT    'Purple' AS           CarColor
UNION    SELECT    'Red' AS   CarColor
UNION    SELECT    'Rose' AS  CarColor
UNION    SELECT    'Rust' AS  CarColor
UNION    SELECT    'Silver' AS           CarColor
UNION    SELECT    'Tan' AS   CarColor
UNION    SELECT    'Turquoise' AS        CarColor
UNION    SELECT    'White' AS            CarColor
UNION    SELECT    'Yellow' AS CarColor
UNION    SELECT    'Multiple' AS CarColor) AS Colors
LEFT JOIN (select
CASE
   WHEN ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=s.VIN) LIKE '%|%' THEN 'Multiple'
  ELSE ( SELECT DISTINCT STRING_AGG(c.Color,' | ') FROM Color c WHERE c.VIN=s.VIN)  END AS ColorCase ,s.Sale_date
from Sale s ) as s ON CarColor = ColorCase GROUP BY CarColor
```

Sales By Type:

Abstract Code

- Clicking **Sales by Type**- Jump to **Sales by Type** task.
    - User (Manager/Roland) selects **Sales by Type** from the drop down **Generate/View reports** menu from **View Reports** form.
    - Run query on the **SALE, VEHICLE** tables
    - Using **VIN** as a join and Checking where sold price is not null
        - For each vehicle type (subtype):
            - Count all sale where *Sale_date* < current date -30 (for the last month's sales)
                - Display vehicle type and count for each type and if no row with sold price not null ,display 0.

get the last available sale date into $MaxSaleDate

```
SELECT MAX(Sale_date) from Sale
```

```
SELECT VT
,count(CASE WHEN sales.Sale_date > $MaxSaleDate -30 THEN sales.Sale_date ELSE NULL END) lastMonth
,count(CASE WHEN sales.Sale_date > $MaxSaleDate -365 THEN sales.Sale_date ELSE NULL END) LastYear
,count(sales.Sale_date) Alltime
from (
select 'Car' as VT
UNION select 'SUV' as VT
UNION select 'Truck' as VT
UNION select 'Convertible' as VT
UNION select 'VanMinivan' as VT
) as UnionVt
LEFT JOIN (
SELECT v.VIN,VehicleType.Vehicle_type,s.Sale_date
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION    SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION    SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION    SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION    SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
LEFT JOIN Sale s ON V.VIN=s.VIN
WHERE s.Sale_date is not null
) AS sales ON sales.Vehicle_type = UnionVt.VT
GROUP BY UnionVt.VT
ORDER BY VT ASC
```

Sales By Manufacturer:

Abstract Code

- Clicking **Sales by Manufacturer**- Jump **to Sales by Manufacturer** task.
    - User (Manager/Roland) selects **Sales by Manufacturer** from the drop down **Generate/View reports** menu from **View Reports** form.
    - Perform query on **SALE,VEHICLE** tables
    - Using *VIN* as a join and checking *Sale_price* is not null to get vehicles that have been sold.
    - for each *Manufacturer_name*:
        - count all sale where *Sale_date* > current date -30 (for the last month's sales), display *Manufacturer_name* and count

get the last available sale date into $MaxSaleDate

```
SELECT MAX(Sale_date) from Sale
```

```
SELECT (Manufacturer_name)
,count(CASE WHEN Sale.Sale_date > $MaxSaleDate -30 THEN Sale.Sale_date ELSE NULL END) LastMonth
,count(CASE WHEN Sale.Sale_date >$MaxSaleDate -365 THEN Sale.Sale_date ELSE NULL END) LastYear
,count(Vehicle.VIN) Alltime
FROM Vehicle LEFT OUTER JOIN Sale ON Vehicle.VIN = Sale.VIN
WHERE Sale.Sale_price IS NOT NULL
GROUP BY Manufacturer_name
ORDER BY  Manufacturer_name  ASC
```

Gross Customer Income:

Abstract Code

- Clicking **Gross Customer Income**- Jump to **Gross Customer Income** task.
    - Perform query of the **SALE**, **VEHICLE,CUSTOMER** and **REPAIR** tables, order by gross income descending and last sale/repair data descending then get the top 15:
        - First, join **CUSTOMER**, **SALE**, and **REPAIR** tables on **CUSTOMER** key attributes.
            - Select *Name* and *Business_name, Sales_price, Sale_date, Start_date*, *Completion_date*, *Total_cost.*
            - Each customer will either have a sale, a repair, or both.
        - Group by **CUSTOMER** key attributes and for each customer/group:
            - include only repairs in progress (WHERE *Completion_date* = Null and *Start_date* is not Null)
            - Get first sale/repair
                - order ascending by *Sale_date* select first row *Sale_date* field
                - order ascending by *Start_date* and select first row *Start_date* field
            - Get most recent sale/repair
                - order descending by *Sale_date* select first row *Sale_date* field
                - order descending by *Start_date* and select first row *Start_date* field
            - number of sales/repairs:
                - Count rows of resulting sales
                - count rows of resulting repairs
            - Gross income:
                - Sum all vehicle sales
                - Sum all *Total_costs* for repairs
                - Add the sales and total repair costs together for each customer to get the total revenue from the customer
    - Final fields in result: *Name* (full) or *Business_name*, date of first sale/repair start, date of most recent sale/repair start, number of sales/repairs, gross income (sales and total repairs costs combined)

```
SELECT TOP 15
CP.CustomerName
,CASE
  WHEN MIN(s.Sale_date) IS NULL AND MIN(r.Start_date) IS NULL THEN NULL
  WHEN MIN(s.Sale_date) IS NULL AND MIN(r.Start_date) IS NOT NULL THEN MIN(r.Start_date)
  WHEN MIN(s.Sale_date) IS NOT NULL AND MIN(r.Start_date) IS NULL THEN MIN(s.Sale_date)
  WHEN MIN(s.Sale_date) > MIN(r.Start_date) THEN MIN(r.Start_date)
  ELSE MIN(s.Sale_date)
END AS  FirstService
,CASE
  WHEN MAX(s.Sale_date) IS NULL AND MAX(r.Start_date) IS NULL THEN NULL
  WHEN MAX(s.Sale_date) IS NULL AND MAX(r.Start_date) IS NOT NULL THEN MAX(r.Start_date)
```

```
  WHEN MAX(s.Sale_date) IS NOT NULL AND MAX(r.Start_date) IS NULL THEN MAX(s.Sale_date)
  WHEN MAX(s.Sale_date) > MAX(r.Start_date) THEN MAX(s.Sale_date)
  ELSE MAX(r.Start_date)
END AS MostRecentService
,COUNT(s.Sale_date) AS NumberOfSales
,COUNT(r.Start_date) AS NumberOfRepairs
,ISNULL(SUM(s.Sale_price),0)  + ISNULL(SUM(r.Total_cost),0) AS GrossIncome
FROM Customer c
LEFT JOIN (SELECT p.Customer_id, (p.First_name + ' ' + p.Last_name)as CustomerName FROM Person p
                                UNION
                                SELECT b.Customer_id, b.Business_name as CustomerName FROM Business b) AS  CP
    ON c.Customer_id = CP.Customer_id
LEFT JOIN Sale s ON c.Customer_id = s.Customer_id
LEFT JOIN Repair r ON c.Customer_id = r.Customer_id AND s.VIN=r.VIN
GROUP BY C.Customer_id,CP.CustomerName
ORDER BY GrossIncome DESC, MostRecentService DESC
```

Clicking **Customer Name** on the Gross Customer Income report- Jump to **Customer Drill-Down** report.

- Save the value of the clicked customer name as variable *$selectedCustomer*
- Customer-Drill-Down Query:
  - Vehicle Sales part of the report will follow query:
    - Join **CUSTOMER, SALE PEOPLE , VEHICLE, EMPLOYEE**
    - Select *VIN*, *Year*, *Manufacturer_name*, *Model_name*
    - if *$selectedCustomer has first and last name, Query the **Person** table.*
    - if *$selectedCustomer* has only Business name, query **Business** table.
    - select all where customers name == *$selectedCustomer*.
    - include fields: *Sale_date*, *Sale_price*, VIN, *Year*, *Manufacturer_name*, *Model_name*, and salesperson *Name*
    - order by Sale_date descending and by *VIN* ascending

Sales Section

```
SELECT
CP.CustomerName,s.Sale_date
,s.Sale_price
,s.VIN
,v.Year
,v.Manufacturer_name
,v.Model_name
,eu.First_name + ' ' + eu.Last_name AS SalesPersonName
FROM Customer c
LEFT JOIN (SELECT p.Customer_id, (p.First_name + ' ' + p.Last_name)as CustomerName FROM Person p
                                UNION
                                SELECT b.Customer_id, b.Business_name as CustomerName FROM Business b) AS  CP
    ON c.Customer_id = CP.Customer_id
LEFT JOIN Sale s ON c.Customer_id = s.Customer_id
LEFT JOIN Vehicle v ON s.VIN = v.VIN
LEFT JOIN EmployeeUser eu ON s.Username = eu.Username
WHERE c.Customer_id = '$Customer_id'
ORDER BY s.Sale_date DESC, s.VIN DESC
```

- Repairs part of the report will follow query:
    - Select *VIN, Start_date, Completion_date*, *Odometer_reading*, *Total_costs*, *Labor_costs*, Service Writer *Name*, Total_parts_cost = (Total_costs - Labor_charges)
    - include fields for each repair: *Start_date*, *Completion_date* (null if repair not finished), *VIN*, *Odometer_reading*, parts *Costs*, *Labor_charges*, *Total_cost*, service writer *Name*.
    - To fulfill this condition “incomplete repairs listed before complete ones with same sorting sort first by end date then start date. “order by *Completion_date* descending, *Start_date* descending, and *VIN* ascending.

Repairs Section

```
SELECT
CP.CustomerName
,r.Start_date
,r.Completion_date
,r.VIN
,r.Odometer_reading
,r.Labor_charges
,r.Total_cost
,eu.First_name + ' ' + eu.Last_name AS SalesPersonName
FROM Customer c
LEFT JOIN (SELECT p.Customer_id, (p.First_name + ' ' + p.Last_name)as CustomerName FROM Person p
                                UNION
                                SELECT b.Customer_id, b.Business_name as CustomerName FROM Business b) AS  CP
    ON c.Customer_id = CP.Customer_id
LEFT JOIN Repair r ON c.Customer_id = r.Customer_id
LEFT JOIN EmployeeUser eu ON r.Username = eu.Username
WHERE c.Customer_id = '$Customer_id'
ORDER BY r.Start_date DESC, r.VIN DESC, r.Completion_date ASC
```

Average Time In Inventory:

Abstract Code

- Clicking **Average Time in Inventory**- Jump to **Average Time in Inventory** task.
    - User (Manager/Roland) selects **Average Time in Inventory** from the drop down **Generate/View reports** menu.
    - Upon:
    - Clicking **Average Time in Inventory**- jump to **Average Time in Inventory** task.
    - Perform query on **SALE** and **VEHICLE** table for information about the average time a vehicle remains in inventory.
        - Calculate Time_in_Inventory by subtracting *Inventory_date* from *Sale_date*
        - Group by vehicle type and perform AVG function on Time_in_Inventory to get Average_Time_in_Inventory.
        - Replace Null Value in average time as “N/A”
            - Display the result

```
SELECT VT
,ISNULL(CAST(AVG(DATEDIFF(DAY,sales.Inventory_date,sales.Sale_date)+1 )AS varchar),'N/A') AS AVERAGE
from (
select 'Car' as VT
UNION select 'SUV' as VT
UNION select 'Truck' as VT
UNION select 'Convertible' as VT
UNION select 'VanMinivan' as VT
) as UnionVt
LEFT JOIN (
SELECT v.VIN,VehicleType.Vehicle_type,s.Sale_date,v.Inventory_date
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION     SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION     SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION     SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION     SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
LEFT JOIN Sale s ON V.VIN=s.VIN
WHERE s.Sale_date is not null
) AS sales ON sales.Vehicle_type = UnionVt.VT
GROUP BY UnionVt.VT
ORDER BY VT ASC
```

Part Statistics:

Abstract Code

Clicking **Parts Statistics**- Jump to **Parts Statistics** task.

- User (Manager/Roland) selects **Parts Statistics** from the drop down **Generate/View reports** menu.
- Upon:
- Clicking Part Statistics- jump to Parts Statistics task.
- Perform query on **PART** table for information about the parts.
- Select Columns: *Vendor_name, Quantity and Price*
- On **PART** table, group by *Vendor_name*
    - Calculate the Total_Number_of Parts by adding the Quantity
    - Calculate the Total_Dollar_Amount by multiplying the *Quantity* and the *Price*
    - Display the result

```
SELECT Vendor_name,
SUM (Part.price * Part.Quantity) AS [Total_Dollar_Amount], SUM (Quantity) AS [Total_Number_Parts]
FROM Part
GROUP BY Vendor_name
ORDER BY [Total_Dollar_Amount] DESC;
```

Below Cost Sales:

Abstract Code

- For each **SALE**
    - Get **VEHICLE** associated with the **SALE** by *VIN*
        - If the *Sale_price* is less than the *Invoice_price* of the **VEHICLE**
            - Get the **CUSTOMER** associated with each Sale
                - If **CUSTOMER** is an individual **PERSON**
                    - Set $customerName equal to the concatenation of *First_name* and *Last_name*
                - If **CUSTOMER** is a **BUSINESS**
                    - Set $customerName equal to the *Business_name*
            - Get the **USER** associated with each **SALE**
            - Set $soldInvoiceRatio to *Sale_price* divided by *Invoice_price* times 100
            - Display *Invoice_price*, *Sale_price*, $soldInvoiceRatio, $customerName, and *Name of Salesperson*
            - If $soldInvoiceRatio is less than 95
                - Set background to red
- Order by *Sale_date* and $soldInvoiceRatio descending
- The query also returns a div that can style a row in red if Sold_invoice_ratio <95%

```
SELECT
FORMAT (s.Sale_date, 'MM-dd-yyyy') Sale_Date
,v.Invoice_price
,s.Sale_price
,(100*(Sale_price)/(Invoice_price)) AS Profit_ratio
,CASE
   WHEN (100*(Sale_price)/(Invoice_price)) <= 95 THEN 'Red'
 END AS Background
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION    SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION    SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION    SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION    SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
LEFT JOIN Sale s ON v.VIN=s.VIN
LEFT JOIN EmployeeUser e ON s.Username=e.Username
LEFT JOIN EmployeeUser ic ON v.Username=ic.Username
LEFT JOIN Customer c ON s.Customer_id=c.Customer_id
LEFT JOIN (SELECT p.Customer_id, (p.First_name + ' ' + p.Last_name)as CustomerName FROM Person p
                                    UNION
                                    SELECT b.Customer_id, b.Business_name as CustomerName FROM Business b) as CP
            ON CP.customer_id=c.customer_id
  WHERE s.Sale_date IS NOT NULL
ORDER BY
s.Sale_date DESC,
(100*(Sale_price)/(Invoice_price)) DESC
```

Repairs by Manufacturer/Type/Model

Abstract Code

- Query **VEHICLE** and **REPAIR** table
  - Left join **REPAIR** and **PART** tables
  - Group by *VIN* and *Start_date*
    - Sum *Quantity* times *Price* as $partsCost
  - Left join **VEHICLE** and **REPAIR** tables
  - Join **VEHICLE** and **MANUFACTURER** tables
  - Group by *Manufacturer_name*
    - Count *Start_date* as $countRepairs
    - Sum *Labor_charges* as $allLaborCosts
    - Sum $partsCost as $allPartsCosts
    - Sum *Labor_charges* and $partsCost as $totalRepairCosts
  - Order by *Manufacturer_name* ascending
- For each result row
  - Display *Manufacturer_name*, $countRepairs, $allPartsCosts, $allLaborCosts, and $totalRepairCosts

```
SELECT
m.Manufacturer_name
,COUNT(r.Start_date) Repairs
,SUM(p.Quantity * p.Price) AS PartsCost
,SUM(r.Labor_charges) LaborCost
,SUM(r.Total_cost) TotalRepairCost
FROM Manufacturer m
LEFT JOIN Vehicle v ON m.Manufacturer_name = v.Manufacturer_name
LEFT JOIN Repair r ON v.VIN = r.VIN
LEFT JOIN Part p ON r.VIN = p.VIN AND r.Start_date = p.Start_date AND r.Customer_id = p.Customer_id
GROUP BY m.Manufacturer_name
ORDER BY m.Manufacturer_name ASC
```

- If user clicks on row (manufacturer drilldown)

  a total per vehicle type: model repair counts, parts costs, labor costs, and total costs. type and counts, repair count, part costs, labor costs, and total costs.

  - Query **VEHICLE** and **REPAIR** table
  - Inner Join on Vehicle and Repair using VIN as a join and with Manufacturer_name = $selectedManufacturer
    - Group by *Vehicle_type*
      - Count *Start_date* as $countRepairs
      - Sum *Labor_charges* as $allLaborCosts
      - Sum $part Cost as $allPartsCosts

- Sum *Labor_charges* and $partsCost as $totalRepairCosts
- Order by $countRepairs descending

- For each result row
  - Display *Vehicle_type*, $countRepairs, $allPartsCosts, $allLaborCosts, and $totalRepairCosts

Vehicle Type Drilldown: System is going to get $selectedManufacturer from the previous drilldown selection

```
SELECT VT,
   SUM(Labor_charges) AS All_labor_Costs,
   SUM(Total_cost) AS Total_Repair_cost,
   (SUM(Total_cost) - SUM(Labor_charges)) AS All_Parts_Costs,
   COUNT(Start_date) AS Count_Repairs
from (
select 'Car' as VT
UNION select 'SUV' as VT
UNION select 'Truck' as VT
UNION select 'Convertible' as VT
UNION select 'VanMinivan' as VT
) as UnionVt
 JOIN (
SELECT v.VIN,VehicleType.Vehicle_type,v.Model_name,r.Labor_charges,r.Total_cost,r.Start_date
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION   SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION   SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION   SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION   SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
JOIN Repair r ON V.VIN=r.VIN
WHERE Manufacturer_name =  ' $selectedManufacturer'
) AS repairs ON Repairs.Vehicle_type = UnionVt.VT
GROUP BY UnionVt.VT
ORDER BY Count_Repairs ASC;
```

- Query **VEHICLE** and **REPAIR** table
  - Left join **REPAIR** and **PART** tables
  - Group by *VIN* and *Start_date*
    - Sum *Quantity* times *Price* as $partsCost
  - Join **VEHICLE** and **REPAIR** tables
  - Join **VEHICLE** and **MANUFACTURER** tables
  - Where *Manufacturer_name* equals the name on the row selected by the user and *Vehicle_type* equals name on the result row
  - Group by *Model_name*
    - Count *Start_date* as $countRepairs
    - Sum *Labor_charges* as $allLaborCosts
    - Sum $partsCost as $allPartsCosts

        - Sum *Labor_charges* and $partsCost as $totalRepairCosts
    - Order by $countRepairs descending
- For each result row
    - Display *Model_name*, $countRepairs, $allPartsCosts, $allLaborCosts, and $totalRepairCosts

Model Drilldown: System is going to get $selectedManufacturer AND $VehicleType from the previous drilldown select

```
SELECT Model_name,
   SUM(Labor_charges) AS All_labor_Costs,
   SUM(Total_cost) AS Total_Repair_cost,
   (SUM(Labor_charges) - SUM(Total_cost)) AS All_Parts_Costs,
   COUNT(Start_date) AS Count_Repairs
from (
select 'Car' as VT
UNION select 'SUV' as VT
UNION select 'Truck' as VT
UNION select 'Convertible' as VT
UNION select 'VanMinivan' as VT
) as UnionVt
 JOIN (
SELECT v.VIN,VehicleType.Vehicle_type,v.Model_name,r.Labor_charges,r.Total_cost,r.Start_date
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION   SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION   SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION   SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION   SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
 JOIN Repair r ON V.VIN=r.VIN
WHERE Manufacturer_name =  '$selectedManufacturer'
AND VehicleType.Vehicle_type = '$VehicleType'
) AS repairs ON repairs.Vehicle_type = UnionVt.VT
GROUP BY UnionVt.VT,Model_name
ORDER BY Count_Repairs ASC;
```

Monthly Sales

Abstract Code:

When user selects report type Monthly Sales, the report will list year-month stats (group by year and month) and return the follow fields:

1. tot_vehicle_sold - total number of vehicles sold per month,
2. income - total sales income,
3. net_income - total net income or (sold_price-invoice_price),
4. profit_ratio - ratio of sold price and invoice price (sold_price/invoice_price)*100.

The report will have the following characteristics:
Green Row when: sold_price/invoice_price >= 125%
Yellow Row when: sold_price/invoice_price < 110% - yellow row
Order by year and month descending
When reading results on the client side, use column background_color to set a color for the row either green, yellow, or blank based on profit_ratio value.
To generate the report, the client will submit the following query against the database:

```
SELECT
YEAR(s.Sale_date) SaleYear
,MONTH(S.SALE_DATE) SaleMonth
,COUNT(v.VIN) Total_Number_of_Vehicles_Sold
,SUM(s.sale_price) Total_Sales_Income
,SUM(s.Sale_price - v.Invoice_price) total_net_income
,(100*SUM(Sale_price)/SUM(Invoice_price)) AS Profit_ratio
,CASE
   WHEN (100*SUM(Sale_price)/SUM(Invoice_price)) >= 125 THEN 'Green'
   WHEN (100*SUM(Sale_price)/SUM(Invoice_price)) <= 110 THEN 'Yellow'
 END AS Background
FROM Vehicle v
LEFT JOIN (
SELECT Car.VIN, 'Car' AS Vehicle_type FROM Car
UNION    SELECT SUV.VIN, 'SUV' AS Vehicle_type FROM SUV
UNION    SELECT Truck.VIN, 'Truck' AS Vehicle_type FROM Truck
UNION    SELECT Convertible.VIN, 'Convertible' AS Vehicle_type FROM Convertible
UNION    SELECT VanMinivan.VIN, 'VanMinivan' AS Vehicle_type FROM VanMinivan
) AS VehicleType ON v.VIN= vehicleType.VIN
LEFT JOIN Sale s ON v.VIN=s.VIN
LEFT JOIN EmployeeUser e ON s.Username=e.Username
LEFT JOIN EmployeeUser ic ON v.Username=ic.Username
LEFT JOIN Customer c ON s.Customer_id=c.Customer_id
LEFT JOIN (SELECT p.Customer_id, (p.First_name + ' ' + p.Last_name)as CustomerName FROM Person p
                                    UNION
                                    SELECT b.Customer_id, b.Business_name as CustomerName FROM Business b) as CP
            ON CP.customer_id=c.customer_id
  WHERE s.Sale_date IS NOT NULL
GROUP BY
 YEAR(s.Sale_date)
,MONTH(S.SALE_DATE)
ORDER BY
 YEAR(s.Sale_date) DESC
,MONTH(S.SALE_DATE) DESC
```

Monthly Sales Drill Down: When customer click on a specific year-month from the monthly sales report, the user will be taken to a drill down report for the selected year-month with top performing sales people, with the following fields:

First_name
Last_name,
Number_vehicles_sold ,
Total_sales = sum of total sales

Order by Number_vehicles_sold descending, Total_sales descending

when click we get Year and month select on variables $MonthfromDrillDown and $YearfromDrillDown to be used on the drilldown

```
SELECT TOP 1
eu.First_name + ' ' + eu.Last_name AS SalesPersonName
,COUNT(s.Username) NumberVehiclesSold
,YEAR(s.Sale_date) AS SaleYear
,MONTH(s.Sale_date) AS SaleYear
,SUM(s.Sale_price) TotalSales
FROM Sale s
LEFT JOIN EmployeeUser eu ON s.Username = eu.Username
WHERE YEAR(s.Sale_date) = '$YearfromDrillDown' AND MONTH(s.Sale_date) = '$MonthfromDrillDown'
GROUP BY eu.First_name + ' ' + eu.Last_name, YEAR(s.Sale_date) ,MONTH(s.Sale_date)
ORDER BY NumberVehiclesSold DESC, TotalSales DESC
```