Contents

Abstract Code and SQL

Main Menu

- Display the first name, last name, and role of the Privileged User
- Show the ***Search Vehicle*** button, push to go to **Search Vehicle** form
- If the user is a Service Writer:
  - Show the ***View/Add/Update Repair*** button, push to go to **View/Add/Update Repair** task
- If the user is an Inventory Clerk:
  - Show the ***Add Vehicle*** button, push to go to **Add Vehicle** task
- If the user is a Manager:
  - Show the ***Sales by Color Report*** button to the **View Sales by Color Report** task
  - Show the ***Sales by Type Report*** button to the **View Sales by Type Report** task
  - Show the ***Sales by Manufacturer Report*** button to the **View Sales by Manufacturer Report** task
  - Show the ***Gross Customer Income Report*** button to the **View Gross Customer Income Report** task
  - Show the ***Repairs Report*** button to the **View Repairs by Manufacturer/Type/Model Report** task
  - Show the ***Below Cost Sales Report*** button to the **View Below Cost Sales Report** task
  - Show the ***Average Time in Inventory Report*** button to the **View Average Time in Inventory Report** task
  - Show the ***Parts Statistics Report*** button to the **Parts Statistics Report** task
  - Show the ***Monthly Sales Report*** button to the **View Monthly Sales Report** task
- If the user is the Owner:
  - All buttons above will be shown
- Push ***Cancel*** to return to close the page

```
-- Find the role of the user
-- Owner table is not necessary because the owner's username is in all the table
WITH ManagersRole AS
	(SELECT Username, 'Manager'  AS Role FROM Managers),
InventoryClkerksRole AS
	(SELECT Username, 'InventoryClerk'  AS Role, FROM InventoryClkerks),
SalespeopleRole AS
	(SELECT Username, 'Salesperson' AS Role FROM Salespeople),
ServiceWritersRole AS
	(SELECT Username, 'ServiceWriter' AS Role FROM ServiceWriters)
SELECT Username, R.Role FROM PreviledgedUsers AS PU
JOIN (ManagersRole UNION InventoryClkerksRole
	UNION SalespeopleRole UNION ServiceWritersRole) AS R
ON PU.Username = R.Username
WHERE Username = '$Username';
```

Login

- User clicks ***Sign in*** button on **Vehicle Search** form.
  - User enters *username (*‘$UserID’*), password (*‘$Password’*)* in the input fields and clicks ***Enter*** button.

```
SELECT Password
FROM PriviledgedUsers WHERE Username = '$Username';
```

- If *email* and *password* inputs are invalid:
  - Raise message “Invalid input”
- Else, the *username* and *password* inputs are valid:
  - If User record is found but the password != ‘$Password’:
    - Go back to **Login** form, with an error message “Wrong password”.
  - Else:
    - Store login information as session variable ‘$UserID’.
    - **Vehicle Search** Form page shows the login information
    - Go to the **Main Menu**

Search Vehicle

(Subtask: **search vehicle and view result**)

- Display the total number of unsold Vehicles available for purchase

```
SELECT COUNT(*) AS TotalVehiclesForSale FROM Vehicles
INNER JOIN SalesEvents ON SalesEvents.Vin = Vehicles.Vin;
```

- User optionally enters *keyword* (‘$Keyword’) input field;
- User optionally enters *price* (‘$Price’) input field;
- User optionally selects an option from dropdown menus for “Vehicle type”, “Manufacturer”, “Model year”, “Color”

```
-- Create drop down menus
SELECT ManufacturerName FROM Manufacturer;

SELECT Color FROM Colors;

-- Since Type is not allowed to saved as an attribute, create the type list by hand
SELECT 'Car' AS Type UNION SELECT 'SUV' UNION SELECT 'Truck'
UNION SELECT 'Convertible' UNION SELECT 'Van MiniVan';

SELECT DISTINCT ModelYear FROM Vehicles
ORDER BY ModelYear DESC;
```

- Show a ***Sign in*** button to the **Login** task
  - If the user is a Privileged user
  - User enters VIN (‘$VIN’) input field.

- If the Privileged User is a Manager or Owner:

```
-- Check if the user is a manager, assign a boolean variable '$IsManager'
SELECT '$Username'  IN (SELECT Username FROM managers);
```

    - The user has the option to filter by sold vehicles, unsold vehicles, or all vehicles.
- Clicked the ***Search*** button;
- If find Vehicles meet the search criteria;
    - Go to the **Search Result Page** form.
        - Display VIN, Vehicle type, Model Year, Manufacturer, Model, color(s) and List price of all unsold Vehicles match the criteria.
        - Sort the result by VIN ascending

```
-- First, assign the “type” to vehicles; Second, search the table with the given possible inputs '$Type',
-- '$Manufacturer', '$ModelYear', '$MaxPrice', '$MinPrice', '$Color', or  '$Keyword', get “Search Result” subquery
--  “CASE WHEN” statement is added to check if the searching condition is assigned, if not, pass a TRUE
-- Thrid: use the “CASE WHEN THEN…ELSE” to consider three situations in one query:
-- a. Non-login user: show the unsold results;
-- b. login user except manager and owner: show unsold result and allow to search by Vin;
-- c. Manager or owner: show the sold/unsold/all result and allow to search by Vin
-- managers or owner can filter the result with variable '$filter': 'Sold', 'Unsold', or NULL


With TypeInfo AS (
        SELECT Vin, 'Car' AS Type FROM Cars
        UNION SELECT Vin, 'SUV' AS Type FROM SUVs
        UNION SELECT Vin, 'Van MiniVan' AS Type FROM VanMiniVans
        UNION SELECT Vin, 'Truck' AS Type FROM Trucks
        UNION SELECT Vin, 'Convertible' AS Type FROM Convertibles
),
SearchResult AS
(
SELECT Vehicles.Vin, T.Type, ModelYear,  Manufacturer, ModelName AS Model,
        C.VColors AS Colors, InvoicePrice * 1.25 AS ListPrice, S.SaleDate
FROM Vehicles
LEFT OUTER JOIN
        (SELECT Vin, GROUP_CONCAT(Colors SEPARATOR ' ') AS VColors
        FROM VehicleColors GROUP BY Vin
        ) AS C ON C.Vin = Vehicles.Vin
LEFT OUTER JOIN SalesEvents AS S ON S.Vin = Vehicles.Vin
JOIN TypeInfo AS T ON T.Vin =Vehicles.Vin
WHERE TRUE AND
        (CASE WHEN '$Type' IS NOT NULL
        THEN Type = '$Type' ELSE TRUE END)
AND
        (CASE WHEN '$Manufacturer' IS NOT NULL
        THEN Manufacturer = '$Manufacturer' ELSE TRUE END)
AND
        (CASE WHEN '$ModelYear' IS NOT NULL
        THEN ModelYear = '$ModelYear' ELSE TRUE END)
AND
        (CASE WHEN '$MaxPrice' IS NOT NULL
        THEN ListPrice <=  '$MaxPrice'  ELSE TRUE END)
AND
        (CASE WHEN '$MinPrice' IS NOT NULL
        THEN ListPrice >=  '$MinPrice' ELSE TRUE END)
AND
        (CASE WHEN '$Color' IS NOT NULL
        THEN  '$Color' IN C.VColor ELSE TRUE END)
AND
        (CASE WHEN '$Keyword' IS NOT NULL
        THEN  Manufacturer LIKE '%$Keyword%' OR Model LIKE '%$Keyword%'
        OR ModelYear LIKE '%$Keyword%' OR Description LIKE '%$Keyword%'
        ELSE TRUE END)
)
CASE WHEN '$Username'  = IS NULL
        THEN SELECT Vin, Type, ModelYear,  Manufacturer, Model, Colors, ListPrice
        FROM SearchResult
        WHERE SaleDate IS NULL
WHEN '$Username'  = IS NOT NULL AND '$IsManager'  = FALSE
        TEHN SELECT Vin, Type, ModelYear,  Manufacturer, Model, Colors, ListPrice
        FROM SearchResult
        WHERE SaleDate IS NULL
        AND (CASE WHEN  '$Vin' IS NOT NULL
                  THEN Vin = '$Vin'  ELSE TRUE END)
ELSE
        SELECT Vin, Type, ModelYear,  Manufacturer, Model, Colors, ListPrice
        FROM SearchResult
        WHERE TRUE AND
                  (CASE WHEN  '$filter' = 'Sold'
                  THEN SaleDate IS NOT NULL
                  WHEN '$filter' = 'Unsold'
                  THEN SaleDate IS NULL
                  ELSE TRUE END)
        AND
                  (CASE WHEN  '$Vin' IS NOT NULL
                  THEN Vin = '$Vin'
                  ELSE TRUE END)
END
ORDER BY Vin
AS MainTable;

--Save it to MainTable for the check mark in box, which cannot be written in one query
```

- Indicate if the keyword matches the description on a check box X mark.

```
-- select which vehicles need to check mark to show the descrption matched
SELECT Vin FROM MainTable
INNER JOIN Vechiles ON Vechiles.Vin = MainTable.Vin
WHERE Vechiles.Description LIKE '%$Keyword%' ;
```

- If no Vehicles meet the search criteria;
  - An error message “Sorry, it looks like we don’t have that in stock!” is displayed.

- Go back to the **<u>Search Vehicle</u>** form.

(Subtask: **view vehicle details**)

- User Clicks on an individual search result and display the following information in the **<u>Search Vehicle Detail Page</u>** form:
  - Display VIN, vehicle type, attributes for that vehicle type, Model Year, Model Name, Manufacturer, color(s), list price, and description.

```
-- Assume the Vin of the vehicles is '$Vin'
-- 1. Develop the subquery for the vehicle’s information except for the type and the type’s attribute called Details
-- 2. Join the Details table with the specific attribute table of the its type

With Details AS
(
        SELECT V.Vin, ModelYear, ModelName AS Model, Manufacturer,
                1.25*InovicePrice AS ListPrice, C.VColors AS Color, Description,
        FROM  Vehicles AS V
        LEFT OUTER JOIN
                (SELECT Vin, GROUP_CONCAT(Colors SEPARATOR ' ') AS VColors
                FROM VehicleColors GROUP BY Vin
                ) AS C ON C.Vin = V.Vin
        WHERE V.Vin = '$Vin';
)
CASE WHEN '$Vin' IN (SELECT Vin FROM Cars )
        THEN SELECT Details.*, 'Car' AS Type, CA.NumOfDoors FROM Details
        INNER JOIN Cars AS CA ON CA.Vin = Details.Vin
WHEN '$Vin' IN (SELECT Vin FROM SUVs )
        THEN SELECT Details.*, 'SUV' AS Type, SU.DrivetrainType, SU.NunberOfCupholders
        FROM Details INNER JOIN SUVs AS SU ON SU.Vin = Details.Vin
WHEN '$Vin' IN (SELECT Vin FROM VanMiniVans )
        THEN SELECT Details.*, 'Van MiniVan' AS Type, VA.HasDriverSideBackDoor FROM Details
        INNER JOIN VanMiniVans AS VA ON VA.Vin = Details.Vin
WHEN '$Vin' IN (SELECT Vin FROM  Trucks )
        THEN SELECT Details.*, 'Truck' AS Type,  TR.CargoCoverType, TR.NumberOfRearAxles,
                TR.CargoCapacity FROM Details
        INNER JOIN TRUCK AS TR ON TR.Vin = Details.Vin
ELSE
        SELECT Details.*, 'Convertible' AS Type, CO.RoofType, CO.BackSeatCount FROM Details
        INNER JOIN Convertibles AS CO ON CO.Vin = Details.Vin
END
```

- If user is a Salesperson or the Owner:

```
-- Check if the user is a salesperson, if yes we need to show the sale car button
SELECT '$Username'  IN (SELECT Username FROM Salespeople);
```

  - Display a ***Sale Vehicle*** button behind each vehicle to go to the **<u>Sales Order</u>** form
- If user is a Manager or Owner:

```
-- Check if the user is a manager
SELECT '$Username'  IN (SELECT Username FROM Managers);
```

- Find the Inventory Clerk associated with the Vehicle

```
-- get the name of the inventory clerk
SELECT CONCAT(FirstName, ' ', LastName) AS ClerkName
FROM PriviledgedUser
INNER JOIN (SELECT ClerkUsernmae FROM Vehicles WHERE Vin = '$Vin') AS V
ON Username = V.ClerkUsername;
```

- Display the Inventory Clerk's name, the invoice price, and the date added
- If the Vehicle is sold:
  - Find the Sales Event; Get the sale date, sale price
  - Find the Customer associated with the Sales Event; Get the information
  - Find the Salesperson; Get the Salesperson's name
  - Display all the Customer information (except the Key), list price, sale price, sales date, and the Salesperson's 's name.

```
 -- Sales Section
-- Develop a subquery to union the person and business table first, called CustomerInfo
With CustomerInfo AS
(
        SELECT  CONCAT(P.FirstName, ' ', P.LastName) AS Name, NULL AS ContactName,
        NULL AS ContactTitle,C.Phone AS Phone, C.Email AS Email, P.CustomerID,
        CONCAT(C.Street, ', ', C.City, ', ', C.State, ', ', C.Zipcode) AS Address
        FROM Persons AS P
        INNER JOIN Customers AS C ON P.CustomerID = C.CustomerID
UNION ALL
        SELECT Name, (ContactFName, ' ', ContactLName) AS ContactName,
        ContactTitle, C.Phone AS Phone, C.Email AS Email, B.CustomerID,
        CONCAT(C.Street, ', ', C.City, ', ', C.State, ', ', C.Zipcode) AS Address
        FROM Business AS B
        INNER JOIN Customers AS C ON B.CustomerID = C.CustomerID
)
SELECT SalePrice, SaleDate, CONCAT(SA.FirstName, ' ', SA.LastName) AS SalespersonName
CustomerInfo.Name, CustomerInfo.ContactName, CustomerInfo.ContactTitle,
CustomerInfo.Phone, CustomerInfo.Email, CustomerInfo.Address
FROM SalesEvents
INNER JOIN Salespeople AS SP ON SP.Username = SalesEvent.Username
INNER JOIN CustomerInfo ON SalesEvent.CustomerID = CustomerInfo.CustomerID
WHERE SalesEvents.Vin = '$Vin';
```

- If the Vehicle has been repaired:
  - Find the Repair Events; Get the start date, end date, labor cost, part cost, and total cost
  - Find the Customers associated with the Repair Events; Get the Customer's name
  - Find the Service Writer associated with the Repair Events; Get the name
  - Show a "Repair" section, Display the Customers' name, the Service Writer's name, start date, end date, labor cost, part cost, and total cost.

```
-- Repair Section
With CustomerInfo AS
(
	SELECT  CONCAT(P.FirstName, ' ', P.LastName) AS Name , P.CustomerID
	FROM Persons AS P
UNION ALL
	SELECT Name , B.CustomerID
	FROM Business AS B
)
SELECT CustomerInfo.Name, CONCAT(SW.FirstName, ' ', SW.LastName) AS ServiceWriterName,
R.StartDate, R.EndDate, R.Laborchange, IFNULL(PA.PartCost, 0) AS PartCost,
	(IFNULL(PA.PartCost, 0) + R.LaborCharge) AS TotalCostTotalCost
FROM RepairEvents AS R
LEFT JOIN
	(SELECT Vin, StartDate, SUM(QuanityUsed * Price) AS PartCost
	FROM Parts GROUP BY Vin, StartDate)  AS PA
ON R.Vin = PA.Vin AND R. StartDate = PA.StartDate
	WHERE RepairEvents.Vin = '$Vin'
INNER JOIN ServiceWriters AS SW ON SW.Username = R.Username
INNER JOIN CustomerInfo ON R.CustomerID = CustomerInfo.CustomerID;
```

- Push ***Cancel***: Close this page and return to the search vehicle page

Add Vehicle

- Inventory clerk or Owner clicks on the ***Add Vehicle*** button from the **<u>Main Menu</u>** form
- Enter *VIN, manufacturer, model name, model year, invoice price, colors, description*
- Select *type* in the drop-down menu
  - Enter the *specific attribute* of the type

```
--create drop down menus
SELECT ManufacturerName FROM Manufacturer;

SELECT Color FROM Colors;

SELECT 'Car' AS Type UNION SELECT 'SUV' UNION SELECT 'Truck'
UNION SELECT 'Convertible' UNION SELECT 'Van MiniVan';
```

- If information is incomplete or invalid:
  - Raise message “Incomplete or invalid information, please enter again”
- Else:
  - Add a new instance in Vehicles
  - The list price will be 125% * invoice price
  - Save the current date to date added

```
-- check whether the '$ModelYear' is valid
SELECT  '$ModelYear' <= YEAR(GETDATE()) + 1;
-- Create new vehicle tuple
```

```
INTO Vehicles (Vin, Manufacturer, ModelName, ModelYear, DateAdded. InvoicePrice, Description) VALUES
('$Vin', '$Manufacturer', '$ModelName', '$ModelYear', '$DateAdded', '$InvoicePrice', '$Description');

-- Assume the type of the vehicle is '$Type', create new tuple of specific type
CASE WHEN '$Type' = 'Car'
        THEN INSERT INTO Cars (Vin, NumOfDoors) VALUES ('$Vin', '$NumOfDoors')
WHEN '$Type' = 'SUV'
        THEN INSERT INTO SUVs (Vin, DrivetrainType,  NunberOfCupholders
        ) VALUES ('$Vin', '$ DrivetrainType ', '$ NunberOfCupholders')
WHEN '$Type' = 'Van MiniVan'
        THEN INSERT INTO  VanMinVans (HasDriverSideBackDoor) VALUES
        ('$Vin',  '$HasDriverSideBackDoor ')
WHEN '$Type' = 'Convertible'
        THEN INSERT INTO   Convertibles (Vin, RoofType, BackSeatCount) VALUES ('$Vin',
        '$RoofType ', '$ BackSeatCount ')
ELSE
        INSERT INTO Trucks (Vin, CargoCoverType, NumberOfRearAxles, CargoCapacity) VALUES
        ('$Vin', '$CargoCoverType', '$NumberOfRearAxles', '$CargoCapacity')
END;
```

- Return to the <u>**Add Vehicle**</u> form

Search/Add Customer

- User can only access this task by pushing button in <u>**Service Writer**</u> **Form or** <u>**Sales Order Form**</u>

(Subtask: **search customer**)

- User selects Customer type: *person* or *business*, push ***Confirm*** to continue
- If person, enter *license* and push ***Search***:
  - Find the person using the license
  - Display First Name, Last Name, License, Address, Phone Number, Email
- If business, user enter *Tax Identification Number* and push ***Search:***
  - Find the business using the Tax Identification Number
  - Display Name, Tax Identification Number, Contact Name, Contact Title, Address, Phone Number, Email

```
-- Assume '$CustomerType' is either 'Person' or 'Business'
-- Combine the two types of customer into one query
-- 1. Union the person and business table, and filter by customer type '$CustomerType'
-- 2. and id '$Input' for either License or TaxNum

SELECT Name, Id, ContactName, ContactTitle, Address, Phone, Email, CustomerID
FROM
(
        SELECT  CONCAT(P.FirstName, ' ', P.LastName) AS Name, P.License AS Id, C.CustomerID,
        NULL AS ContactName, NULL AS ContactTitle,C.Phone, C.Email,
        CONCAT(C.Street, ', ', C.City, ', ', C.State, ', ', C.Zipcode) AS Address
        FROM Persons AS P
        INNER JOIN Customers AS C ON P.CustomerID = C.CustomerID
UNION ALL
        SELECT Name, TaxNum AS Id, C.CustomerID,
            (ContactFName, ' ', ContactLName) AS ContactName, ContactTitle, C.Phone, C.Email,
            CONCAT(C.Street, ', ', C.City, ', ', C.State, ', ', C.Zipcode) AS Address
            FROM Business AS B
            INNER JOIN Customers AS C ON B.CustomerID = C.CustomerID
)
WHERE CustomerType =  '$CustomerType' AND Id = '$Input'
AS MainTable2
```

- If not found:
    - raise message “Not Found”
- Push ***Search Again*** to return to select customer type
- Push ***Confirm Customer***:
    - Return to the **Service Writer Form** or **Sales Order Form** ; And dispaly “Customer Confirmed” on the fom

```
-- pass the CustomerID to the  Repair/Sale task
SELECT CustomerID FROM MainTable2
```

(Subtask: **add new customer)**

- Push the ***Add New Customer*** button
- User selects Customer type: *person* or *business*
- If Person, enter:
    - *First Name, Last Name, License, Address, Phone Number, Email*
- If Business, enter:
    - *Name, Tax Identification Number, Contact Name, Contact Title, Address, Phone Number, Email*
- Push ***Add*** button:
    - If information is completed and valid:
        - Add a new instance of Customer with the information
    - Else:
        - raise message “Invalid or incomplete information”

```
-- Create new customer No need to insert customer id, which is auto generated
INSERT INTO Customers (Street, City, State, ZipCode, Email, PhoneNum) VALUES ('$Street', '$City', '$State',
'$ZipCode', '$Email', '$PhoneNum');

--get the surrogate key as '$CustomerID', which will be passed to repair/sales task as well
SELECT LAST_INSERT_ID();

--if add a person customer
INSERT INTO Persons (License, FirstName, LastName, CustomerID) VALUES ('$License', '$FirstName',
'$LastName', '$CustomerID');

-- if add a business customer
INSERT INTO Business (TaxNum, Name, ContactFName, ContactLName, ContactTitle, CustomerID) VALUES
('$TaxNum', '$Name', '$ContactFName', '$ContactLName', '$ContactTitle', '$CustomerID');
```

- Push ***Confirm Customer***:
    - Return to the **Service Writer Form** or **Sales Order Form** ; And dispaly “Customer Confirmed” on the fom
- Push ***Cancel*** :

- close the form; return to the **Service Writer Form** or **Sales Order Form** without selecting a customer

Add Sales

- Salespeople push a button on **Search Vehicle Detail Page**
- Find the Vehicle selected in **Search Vehicle** task
- Push a ***Customer*** button to prompt the **Search/Add Customer** form to find **Customer**
- Enter: the *sale price* (‘$salePrice’)
- Push ***Confirm Button***
  - If the ‘$salePrice’ is not valid:
    - raise message “Invalid Entry”
  - If the ‘$SalePrice’ is not higher than 95% of the invoice price, and the user is not Roland Around:
    - raise message “Price too low”

```
-- check the invoice price
SELECT InvoicePrice > 0.95 * '$SalePrice' FROM Vehicles
WHERE Vin = '$Vin';

-- check if the user is the owner
SELECT Username = '$Username'  FROM Owner;
```

  - Else:
    - Update the Sold attribute of Vehicle from Unsold to Sold
    - Add a new Sales Event instance with ‘$salePrice’
    - Save the current date as Sale date of the Sales Event

```
-- Create new sales event, assume Vin is '$Vin', Customer ID is '$CustomerID'
-- salesperson username '$Username', he/she enter the '$SalePrice'
-- Get the date of today as SaleDate
INSERT INTO SalesEvents (Vin, SaleDate, SalePrice, CustomerID, Username) VALUES ('$Vin', (SELECT
GETDATE()), '$SalePrice', '$CustomerID', '$Username');
```

    - Display a message “Sale has been successfully added”
    - Close this **form**; go to the **Search Vehicle Detail Page** form
- Push ***Cancel*** :
  - go to the **Search Vehicle Detail Page** Form without filing the sales

View/Add/Update Repair

- Service Writer or Owner push button ***View/Add/Update Repair*** on the **Main Menu**

(Subtask: **lookup vehicle**)

- User enter the *VIN* (‘$VIN’) and push ***Search*** button,
- If ‘$VIN’ is not found, raise an error message “VIN is not found”.
- If ‘$VIN’ is for an unsold Vehicle, raise message “Unsold Vehicle”
- Else, Display VIN, Vehicle type, Model Year, model name, manufacturer, and colors of the sold Vehicles.

```
-- Check whether the Vin is in the sold vehicles
SELECT ('$Vin' IN (SELECT Vin FROM SalesEvent) );

--If yes, then see the detail of the vehicles
With TypeInfo AS (
	SELECT Vin, 'Car' AS Type FROM Cars
	UNION SELECT Vin, 'SUV' AS Type FROM SUVs
	UNION SELECT Vin, 'Van MiniVan' AS Type FROM VanMiniVans
	UNION SELECT Vin, 'Truck' AS Type FROM Trucks
	UNION SELECT Vin, 'Convertible' AS Type FROM Convertibles
),
SELECT Vehicles.Vin, T.Type, ModelYear,  Manufacturer, ModelName AS Model,
	C.VColors AS Colors
FROM Vehicles
JOIN (SELECT Vin, GROUP_CONCAT(Colors SEPARATOR ' ') AS VColors
	FROM VehicleColors GROUP BY Vin
	) AS C ON C.Vin = Vehicles.Vin
JOIN TypeInfo AS T ON T.Vin =Vehicles.Vin
WHERE  Vehicles.Vin = '$Vin' ;
```

Push ***Search Again*** to go back to search

- Push ***Confirm*** to check the repair event of this vehicle
-

(Subtask: **view open repair event**)

- Find the Open Repair Event associated with the Vehicle
- Find the Needs and Parts associated with Repair Event
- Display the labor cost, total cost, odometer, start date, parts number, parts quantity

```
--Search if there is any incomplete repairs with '$Vin'
SELECT '$Vin' IN (SELECT Vin FROM RepairEvents WHERE EndDate IS NULL);

-- If yes, show the detail
SELECT R.Vin, R.StartDate, R.Odometer, R.LaborCharge, IFNULL(PA.PartCost, 0) AS PartCost,
	(IFNULL(PA.PartCost, 0) + LaborCharge) AS TotalCost
FROM RepairEvents AS R
LEFT JOIN (SELECT Vin, StartDate, SUM(QuanityUsed * Price) AS PartCost
		FROM Parts GROUP BY Vin, StartDate)  AS PA
ON RepairEvents.Vin = PA.Vin AND RepairEvents. StartDate = PA.StartDate
WHERE RepairEvents.Vin = '$Vin' AND EndDate IS NULL;
```

(Subtask: **add a repair event**)

- If no open Repair Event exists, the user push ***Add New Repair***;
  - Push a ***Customer*** button to the **Search/Add Customer** form to find or add Customer
  - The user enters the *odometer*
  - User push ***Add Repair:***
    - If the information is not completed or invalid:
      - Raise a message “Invalid or incomplete information”
    - Else,

        - Add a new Repair Event instance entered information;
        - Save the status of Repair Event is "Open";
        - Save the current date as the start date;

```
-- Subtask create new repair event, assume Vin is '$Vin', Customer ID is '$CustomerID'
-- service writer username '$Username'
INSERT INTO RepairEvents (Vin, StartDate, EndDate, LarborCharge, Odometer, Description, CustomerID,
Username) VALUES ('$Vin', (SELECT GETDATE()), NULL, '0.00', '0', NULL, '$CustomerID', '$Username');
```

    - Push **Continue** Go to **update repair event**
    - Push ***Cancel***, No Repair Event added, display the ***Add New Repair*** button on the form

(Subtask: **update Repair Event)**

- If there is an open Repair Event, Push ***Update*** button
    - User optionally enter a *description* in the input field
    - User optionally enter a new *labor cost* ('$labor cost') in an input field
    - Push ***OK*** button following the input field
        - If '$labor cost' not higher than original and the user is not Owner, or the data is not valid:
            - Raise a message "Invalid entry, Please enter again"

```
-- check the if the new labor charge higher than the old labor charge
SELECT LaborCharge < '$LaborCharge' AS OldCharge
FROM RepairEvents WHERE Vin = '$Vin' AND StartDate = '$StartDate';

-- check if the user is the owner
SELECT Username = '$Username'  FROM Owner;
```

        - Else:
            - Update labor cost, total cost

```
--Substask update repair event, assume '$LaborCharge', '$Odometer', '$Description' are entered by the user
UPDATE RepairEvents
SET LarborCharge = '$LaborCharge', Odometer = '$Odometer', Description = '$Description'
WHERE Vin = '$Vin' AND StartDate = '$StartDate';
```

        - **view open repair event**
    - User optionally enter the *quantity, vendor, part number, price* of Parts in an input field
    - Push ***OK*** button following the input field
        - If the information is valid:
            - Update information

```
--If user add parts '$Vin', '$StartDate' are the primary key of the RepairEvents tuple
--'$Number', '$Price', '$QuantityUsed' are entered by the user
INSERT INTO Parts (Vin, StartDate, Price, Number, QuantityUsed) VALUES ('$Vin', '$StartDate', '$Price',
'$Number', '$QuantityUsed');

--If the user update the quantity used of parts, '$NewQuantityUsed' is entered by the user
UPDATE Parts
SET QuantityUsed = '$NewQuantityUsed'
WHERE Vin = '$Vin' AND StartDate = '$StartDate' AND Number = '$Number';
```

- **view open repair event**
        - Else:
            - raise message “Invalid or incomplete information”
    - Push ***Save for Later***
        - Update information of the Repair Event and the Needs
        - Status of the Repair Event is “Open”.
        - **view open repair event**
    - Push ***Save and Close Repair***
        - Update the Repair Event and Needs
        - Status of Repair Event is “Close”
        - Save the current date of Repair Event as end date

```
--If the close the repair event
UPDATE RepairEvents
SET EndDate = (SELECT GETDATE())
WHERE Vin = '$Vin' AND StartDate = '$StartDate';
```

        - Close the **form** and return to main menu
    - Push ***Cancel***:
        - No Information updated and return to the main menu

View Sales by Color Report

- Page open by clicking on ***Sales by Color Report*** button from the **Main Menu**
- Get the sold date from Sale Events; Get the last sold date in all the Sales Events
- Find the sold Vehicles of each Sales Event; Get the colors of the Vehicles
- Display a table with three columns: “Sales within 30 days”, “Sales within last year”, “All time sales”
    - Aggregate number of sold Vehicles by color, where sold date <= last sold date - 30, as Last 30 days' sales
    - Aggregate number of sold Vehicles by color, where sold date <= last sold date - 365, as Last year sales
    - Aggregate number of all sold Vehicles by color, as All time sales
    - In all the aggregations, if the vehicle colors have multiple values:
        - Aggregate the number to the row “multiple” separately
    - If no Vehicle sold in a color in each aggregation:
        - Assign “0” to the value for that color
- Display the Last 30 days sales in the first column
- Display the Last year sales in the second column
- Display the All time sales color in the third column

```
-- Get  '$LastSaleDate' , same process at the beginning of Sales by type and by manufacturer
-- Actually this can be nested in the main query, but may cuase confusion
SELECT MAX(SaleDate) FROM SalesEvent;

-- 1. Find the Vehicles with single color, and those with multiple colors
-- 2. For the single color vehicles, get the sales cout of month, year, and all time by color
-- 3. Join the result of month year, all time forr single color vehilces
-- 4. Union the single color table with the row "multiple" with the sale of month year and all time

With SingleColorSoldVehicles AS
(
        SELECT SalesEvent.Vin, SaleDate, VehilceColors.Colors AS Color
        FROM SalesEvents
        INNER JOIN VehicleColors ON VehicleColors .Vin = SalesEvents.Vin
        WHERE SalesEvents.Vin IN
        (SELECT VIN FROM
                (Vin, Count(*) AS ColorCount
                FROM VehicleColors
                GROUP BY Vin HAVING ColorCount = 1)
        )
),
MultiColorSoldVehicles AS
(
        SELECT Vin, SaleDate FROM SalesEvents
        WHERE Vin NOT IN
        (SELECT VIN FROM SingleColorSoldVehicles)
),
SingleColorVehicleMonthSalesByColor AS
(
        SELECT Color, Count(*) AS SaleCount FROM SingleColorSoldVehicles
        WHERE '$LastSaleDate' - SaleDate  <= 30
        GROUP BY Color
),
SingleColorVehicleYearSalesByColor AS
(
        SELECT Color, Count(*) AS SaleCount FROM SingleColorSoldVehicles
        WHERE '$LastSaleDate' - SaleDate  <= 365
        GROUP BY Color
),
SingleColorVehicleAllTimeSalesByColor AS
(
        SELECT Color, Count(*) AS SaleCount FROM SingleColorSoldVehicles
        GROUP BY Color
)
SELECT DISTINCT C.Color,
        IFNULL (M.SaleCount,0) AS MonthlySales,
        IFNULL (Y.SaleCount,0) AS YearSales,
        IFNULL (A.SaleCount,0) AS AllTimeSales
FROM Colors AS C
LEFT OUTER JOIN SingleColorVehicleMonthSalesByColor AS M
        ON C.Color = M.Color
LEFT OUTER JOIN SingleColorVehicleYearSalesByColor AS Y
        ON C.Color = Y.Color
LEFT OUTER JOIN SingleColorVehicleAllTimeSalesByColor AS A
        ON V.Color = A.Color
UNION ALL
SELECT 'Multiple' AS Color,
(SELECT COUNT(*) FROM MultiColorSoldVehicles
        WHERE  '$ LastSaleDate' - SaleDate  <= 30)  AS MonthlySales,
(SELECT COUNT(*) FROM MultiColorSoldVehicles
        WHERE '$ LastSaleDate' - SaleDate  <= 365)  AS YearSales,
(SELECT COUNT(*) FROM MultiColorSoldVehicles)  AS AllTimeSales
ORDER BY Color;
```

- Push ***Close*** to return to **<u>Main Menu</u>**

View Sales by Type Report

- Page open by pushing ***Sales by Type Report*** button on **<u>Main Menu</u>**
- Get the sold dates from Sale Events; Get the last sold date in all the Sales Events
- Find the sold Vehicles of each Sales Event; Get the type of each Vehicle
- Display a table with three columns: “Sales within 30 days”, “Sales within last year”; “All time sales”
    - Aggregate number of sold Vehicles by type, where sold date <= last sold date - 30, as <u>Last 30 days' sales</u>
    - Aggregate number of sold Vehicles by type, where sold date <= last sold date - 365, as <u>Last year sales</u>
    - Aggregate number of all sold Vehicles by type, as <u>All time sales</u>
    - If no Vehicle sold in a color in each aggregation:
        - Assign “0” to the value for that type
- Display the <u>Last 30 days sales</u> in the first column
- Display the <u>Last year sales</u> in the second column
- Display the <u>All time sales</u> color in the third column

```
-- Similar to the sales by color
-- 1.Assign type to the vehicles, then develop the VehiclesWithType  subquery
-- 2.Get the sale count of month, year, all time by type
-- 3.Create a column with all types by hand, in case some types have no sales at all
-- 4.Join the month, year, all time sales

With TypeInfo AS (
        SELECT Vin, 'Car' AS Type FROM Cars
        UNION SELECT Vin, 'SUV' AS Type FROM SUVs
        UNION SELECT Vin, 'Van MiniVan' AS Type FROM VanMiniVans
        UNION SELECT Vin, 'Truck' AS Type FROM Trucks
        UNION SELECT Vin, 'Convertible' AS Type FROM Convertibles
),
VehiclesWithType AS
(
        SELECT Vs.Vin, T.Type FROM Vehicles AS Vs
        JOIN TypeInfo AS T ON T.Vin =Vs.Vin
),
TypeSoldVehicles AS
(
        SELECT SalesEvents.Vin, VT.SaleDate, VT.Type FROM SalesEvents
        INNER JOIN VehiclesWithType AS VT ON SalesEvents .Vin = VT.Vin
),
VehicleMonthSalesByType AS
```

```
(
        SELECT Type, Count(*) AS SaleCount FROM TypeSoldVehicles
        WHERE '$LastSaleDate' - SaleDate  <= 30
        GROUP BY Type
),
VehicleYearSalesByType AS
(
        SELECT Type, Count(*) AS SaleCount FROM TypeSoldVehicles
        WHERE '$LastSaleDate' - SaleDate  <= 365
        GROUP BY Color
),
VehicleAllTimeSalesByType AS
(
        SELECT Type, Count(*) AS SaleCount FROM TypeSoldVehicles
        GROUP BY Type
),
AllTypes AS (
        SELECT 'Car' AS Type UNION SELECT 'SUV' UNION SELECT 'Truck'
        UNION SELECT 'Convertible' UNION SELECT 'Van MiniVan'
)
SELECT T.Type,
        IFNULL (M.SaleCount,0) AS MonthlySales,
        IFNULL (Y.SaleCount,0) AS YearSales,
        IFNULL (A.SaleCount,0) AS AllTimeSales
FROM AllTypes AS T
LEFT OUTER JOIN VehicleMonthSalesByType AS M ON T.Type = M.Type
LEFT OUTER JOIN VehicleMonthSalesByType AS Y ON T.Type = Y.Type
LEFT OUTER JOIN VehicleAllTimeSalesByType AS A ON T.Type = A.Type
ORDER BY Type;
```

- Push ***Close*** to return to **Main Menu**

View Sales by Manufacturer Report.

- Page open by pushing ***Sales by Manufacturer Report*** button button on **Main Menu**
- Get the sold dates from sale events; Get the last sold date in all the Sales Events
- Find the sold Vehicles of each Sales Event; Get the manufacturer of each Vehicle
- Display a table with three column: “Sales within 30 days”, “Sales within last year”; “All time sales”
  - Aggregate number of sold Vehicles by manufacturer, where sold date <= last sold date - 30, as Last 30 days sales
  - Aggregate number of sold Vehicles by manufacturer, where sold date <= last sold date - 365, as Last year sales
  - Aggregate number of all sold Vehicles by manufacturer, as All time Sales
  - If no Vehicle sold in a color in each aggregation:
    - Assign “0” to the value for that type
  - If all three values of a manufacturer is “0”:
    - Remove the row of the manufacturer
- Display the Last 30 days sales in the first column
- Display the Last year sales in the second column
- Display the All time Sales color in the third column

```
-- Similar to the two tables above. But need to exclude rows with 0 sale in three columns

With ManuSoldVehicles AS
(
	SELECT SalesEvents.Vin, SaleDate, Vehicles.Manufacturer FROM SalesEvents
	INNER JOIN Vehicles ON Vin
),
VehicleMonthSalesByManu AS
(
	SELECT Manufacturer, Count(*) AS SaleCount FROM ManuSoldVehicles
	WHERE '$LastSaleDate' - SaleDate  <= 30
	GROUP BY Manufacturer
),
VehicleYearSalesByManu AS
(
	SELECT Manufacturer, Count(*) AS SaleCount FROM ManuSoldVehicles
	WHERE '$LastSaleDate' - SaleDate  <= 365
	GROUP BY Manufacturer
),
VehicleAllTimeSalesByManu AS
(
	SELECT Manufacturer, Count(*) AS SaleCount FROM ManuSoldVehicles
	GROUP BY Manufacturer
),
SELECT ManufacturerName,
	IFNULL (M.SaleCount,0) AS MonthlySales,
	IFNULL (Y.SaleCount,0) AS YearSales,
	IFNULL (A.SaleCount,0) AS AllTimeSales
FROM Manufacturer AS Manu
LEFT OUTER JOIN VehicleMonthSalesByManu AS M
	ON Manu.ManufacturerName = M.Manufacturer
LEFT OUTER JOIN VehicleMonthSalesByManu AS Y
	ON Manu.ManufacturerName = Y.Manufacturer
LEFT OUTER JOIN VehicleAllTimeSalesByManu AS A
	 ON Manu.ManufacturerName = A.Manufacturer
WHERE AllTimeSales != 0 OR YearSales != 0 OR MonthSales != 0
ORDER BY ManufacturerName;
```

- Push ***Close*** to return to **<u>Main Menu</u>**

View Gross Customer Income Report

- Page open by pushing ***Gross Customer Income Report*** button on **<u>Main Menu</u>**
- Get the name of Customers:
    - If Person: get first name + last name
    - Else: get Business name

(Subtask: **look up customer sales overview**)

- Find the Sales Events associated with customers; Get the sale price, sold date of the Sales Events
    - Sum the sale price by customer as <u>customer sales income</u>
    - Aggregate the number of sales by customer as <u>customer sales number</u>
    - Get the max and min of sale date of customers

(Subtask: **look up customer repair overview**)

- the Repair Events associated with customers; Get the repair cost, start date of the Repair Events
    - Sum the repair cost by customer as customer repair income
    - Aggregate the number of repairs by customer as customer repair number
    - Get the max and min of repair start date of customers
- Get the total gross incomes by customer repair income + customer sales income
- Get the date of first sale/repair by min(first sold date, first repair date)
- Get the date of last sale/repair by max(last sold date, last repair date)
- Sort the result by total gross incomes descending and by date of last sale/repair descending
- Display the first 15 result of customers' name, date of first sale/repair, date of last sale/repair, total gross incomes in rows

```
-- Create three subqueries: customer info, sales info, repair info
-- Join them on Customer ID

WITH CustomerInfo AS
(
        SELECT CustomerID, CONCAT(FirstName, ' ', LastName) AS Name FROM Persons AS P
        UNION ALL
        SELECT CustomerID, Name FROM Business
)
SalesInfo AS
(
        SELECT CustomerID, SUM(SalePrice) AS SalesIncome, Count(*) AS SalesNumber,
        MAX(SalesDate) AS LastSalesDate, MIN(SalesDate) AS FirstSalesDate
        FROM SalesEvents
        GROUP BY CustomerID
)
RepairInfo AS
(
        SELECT CustomerID, SUM(LaborCharge) + SUM(IFNULL(PA.PartCost,0) ) AS RepairIncome,
        Count(*) AS RepairNumber, MAX(RepairEvents.StartDate) AS LastRepairDate,
        MIN(RepairEvents.StartDate) AS FirstRepairDate
        FROM RepairEvents
        LEFT JOIN
                (SELECT Vin, StartDate, SUM(QuanityUsed * Price) AS PartCost
                FROM Parts GROUP BY Vin, StartDate)  AS PA
        ON RepairEvents.Vin = PA.Vin AND RepairEvents. StartDate = PA.StartDate
        GROUP BY CustomerID
)
SELECT C.Name, MIN(R.FirstRepairDate, S.FirstSalesDate) AS FirstDate,
        MAX(R.LastRepairDate, S.LastSalesDate) AS LastDate, S.SalesNumber, R.RepairNumber,
        IFNULL(R.RepairIncome,0) + IFNULL(S.SalesIncome,0) AS TotalIncome
FROM CustomerInfo AS C
LEFT OUTER JOIN SalesInfo AS S ON C.CustomerID = S.CustomerID
LEFT OUTER JOIN RepairInfo AS R ON C.CustomerID =   R.CustomerID
ORDER BY TotalIncome DESC, LastDate DESC
LIMIT 15;
```

- Push the ***More Detail*** button of a customer to open the drill-down
- Get the name of the selected customer

(Subtask: **Look up customer sales details**)

- Find Sales Events associated with the customer; Get sold date, sale price of Sales Events
- Find Vehicles of the Sales Events: Get VIN, model year, manufacturer, model name,
- Find Salesperson of the Sales Events: Get Salespeople's name
- Display the sale date, sale price, VIN, model year, manufacturer, model name, and Salesperson's name in rows
    - Sort by sale date descending and VIN ascending

```
--Sales section, assume the customer ID is '$ID'
SELECT  SalesDate, SalePrice, V.Vin, V.ModelYear AS YEAR, V.Manufacturer,
        V.ModelName AS Model,  CONCAT(PU.FirstName, ' ', PU.LastName) AS SalespersonName
FROM SalesEvent
LEFT JOIN Vehicles AS V ON V.Vin = SalesEvent.Vin
LEFT JOIN PriviledgedUsers AS PU ON PU.Username = SalesEvent.Username
WHERE SalesEvent.CustomerID = '$ID'
ORDER BY SalesDate DESC, Vin ASC;
```

(Subtask: **Look up customer repair details**)

- Find Repair Events of the customer; get start day, end day, odometer, part cost, labor cost, total cost
- Find the Vehicles associated with the Repair Events: get the VIN
- Find the Service Writer associated with the Repair Event: Get the Service Writer's name
- Display the start date, end date, VIN, odometer, part costs, labor cost, total cost Service Writer's name
    - Sort the result in repair start date descending, end date descending, VIN ascending
    - List incomplete Repair Events first
    - If end date in Null: do not show the end date

```
--Repair section, assume the customer ID is '$ID'
SELECT StartDate, EndDate, Odometer, LaborCharge, IFNULL(PA.PartCost,0) AS PartCost
        (IFNULL(PA.PartCost,0) + LaborCharge) AS TotalCharge,
        CONCAT(PU.FirstName, ' ', PU.LastName) AS ServiceWriterName
FROM RepairEvents
LEFT JOIN
        (SELECT Vin, StartDate, SUM(QuanityUsed * Price) AS PartCost
        FROM Parts GROUP BY Vin, StartDate)  AS PA
ON RepairEvents.Vin = PA.Vin AND RepairEvents. StartDate = PA.StartDate
LEFT JOIN PriviledgedUsers AS PU ON PU.Username = RepairEvents.Username
WHERE RepairEvents.CustomerID = '$ID'
ORDER BY StartDate DESC, EndDate IS NULL, EndDate DESC, Vin ASC;
```

- Push ***Cancel*** to close the drill-down
- Push ***Close*** to return to **Main Menu**

View Repairs by Manufacturer/Type/Model Report

- Page open by pushing ***Repairs by Manufacturer/Type/Model Report*** button on **Main Menu**

(Subtask: **Look up Repair Overview**)

- Get the manufacturer, types, models of Vehicles
- Find Repair Events for the Vehicles; Get the total cost, labor cost, part cost in Repair Events
  - Aggregate the count of repairs, total cost, labor cost, and part cost by Vehicle
  - Aggregate the results of Vehicle by manufacturer as manufacturer repair count, manufacturer total cost, manufacturer labor cost, manufacturer part cost
- Sort the result by manufacturer ascending
- Display manufacturer repair count, manufacturer total cost, manufacturer labor cost, manufacturer part cost
- Include manufacturers if do not have records

```
-- Left out join to include all manufacturer


WITH RepairInfo AS
(
        SELECT RepairEvents.Vin, StartDate, V.Manufacturer, LaborCharge,
                IFNULL(PA.PartCost, 0) AS PartCost,
                (IFNULL(PA.PartCost, 0) + LaborCharge) AS PartLaborCost
        FROM RepairEvents
        LEFT JOIN
                (SELECT Vin, StartDate, SUM(QuanityUsed * Price) AS PartCost
                FROM Parts GROUP BY Vin, StartDate)  AS PA
        ON RepairEvents.Vin = PA.Vin AND RepairEvents. StartDate = PA.StartDate
        LEFT JOIN Vehicles AS V ON RepairEvents.Vin = V.Vin
)
SELECT M.ManufacturerName, RR.TotalRepairCount, RR.TotalLaborCost, RR.TotalPartCost,
        RR.TotalLaborPartCost
FROM Manufacturer AS M
LEFT OUTER JOIN
        (Manufacturer, Count(*) AS TotalRepairCount,
        SUM(LaborCharge) AS TotalLaborCost, SUM(PartCost) AS TotalPartCost,
        SUM(PartLaborCost) AS TotalLaborPartCost
        FROM RepairInfo
        GROUP BY Manufacturer) AS RR
ON M.ManufacturerName =  RR.Manufacturer
ORDER BY ManufacturerName;
```

(Subtask: **Look up Repair Details**)

- Push the ***More Detail*** button of each row to open drill down
- Find the Vehicles in the in manufacturer selected; Get the types and models of the Vehicles
  - Aggregate the results of Vehicle by type as type repair count, type total cost, type labor cost, type part cost
- Sort the result by type repair count descending
  - Aggregate the results of Vehicle by model as model repair count, model total cost, model labor cost, model part cost
  - Sort the result by model repair count descending of each type

- Display type repair count, type total cost, type labor cost, type part cost in rows
- Exclude types if do not have record
- After each type row, display model repair count, model total cost, model labor cost, model part cost
- Exclude types if do not have record

```
-- assume the selected manufacturer is '$Manu'
-- 1. Assign types to vehicles
-- 2. Create a rerpair info table for each repair event
-- 3. Union by-type and by-type-model

With TypeInfo AS (
        SELECT Vin, 'Car' AS Type FROM Cars
        UNION SELECT Vin, 'SUV' AS Type FROM SUVs
        UNION SELECT Vin, 'Van MiniVan' AS Type FROM VanMiniVans
        UNION SELECT Vin, 'Truck' AS Type FROM Trucks
        UNION SELECT Vin, 'Convertible' AS Type FROM Convertibles
),
VehiclesWithType AS
(
        SELECT Vs.Vin, ModelName, ModelYear, DateAdded, InvoicePrice,Manufacturer,
        ClerkUsername, Decription , T.Type FROM Vehicles AS Vs
        JOIN TypeInfo AS T ON T.Vin =Vs.Vin
),
RepairInfo AS
(
        SELECT RepairEvents.Vin, StartDate, LaborCharge, IFNULL(PA.PartCost, 0) AS PartCost
                (IFNULL(PA.PartCost, 0) + LaborCharge) AS PartLaborCost,
                V.Type, V.Manufacturer, V.ModelName AS Model
        FROM RepairEvents
        LEFT JOIN
                (SELECT Vin, StartDate, SUM(QuanityUsed * Price) AS PartCost
                FROM Parts GROUP BY Vin, StartDate)  AS PA
        ON RepairEvents.Vin = PA.Vin AND RepairEvents. StartDate = PA.StartDate
        JOIN VehiclesWithType AS V ON RepairEvent.Vin = V.Vin
)
SELECT Type, NULL AS Model, Count(*) AS TotalRepairCount,
        SUM(LaborCharge) AS TotalLaborCost, SUM(PartCost) AS TotalPartCost,
        SUM(PartLaborCost) AS TotalPartLaborCost
FROM RepairInfo
WHERE ManufacturerName = '$Manu'
GROUP BY Type
UNION ALL
SELECT Type, ModelName AS Model, Count(*) AS TotalRepairCount,
        SUM(LaborCharge) AS TotalLaborCost, SUM(PartCost) AS TotalPartCost,
        SUM(PartLaborCost) AS TotalPartLaborCost
FROM RepairInfo
WHERE ManufacturerName = '$Manu'
GROUP BY Type, Model
ORDER BY Type ASC, Model IS NULL, Model ASC;
```

- Push ***Cancel*** to close the drill-down
- Push ***Close*** to return to **Main Menu**

View Below Cost Sales Report

- Page open by pushing ***Below Cost Sales Report*** button on **Main Menu**

(Subtask: **view below cost sales**)

- Get sale price and sale date in Sales Events
- Find Vehicles associated with Sales Events; Get invoice price of Vehicles
- Select the Vehicles sold below cost by where(Vehicle.inovice price >Sales Event.sale price);
  - Get the price ratio by sale price/invoice price

(Subtask: **view detail of below cost sale**)

- Find Customer for the selected Vehicle
- If the customer is a person:
  - get first name and last name
- Else:
  - get business name
- Find Salesperson for the selected Vehicles; Get the name of the Salesperson
- Sort the result by sale date descending and price ratio descending
- Display sale date, sale price, price ratio, customer name, Salesperson name in rows

```
SELECT S.SaleDate, S.SalePrice, V.InvoicePrice, (S.SalePrice / V.InvoicePrice) AS PriceRatio,
        C.Name AS CustomerName , COTCAT(P.FirstName, ' ', P.LastName) AS SalespersonName
FROM SalesEvents AS S
INNER JOIN Vehicles AS V ON V.Vin = S.Vin
INNER JOIN
(
        SELECT CustomerID, CONCAT(FirstName, ' ', LastName) AS Name FROM Persons
UNION ALL
        SELECT CustomerID, Name FROM Business
) AS C ON C.CustomerID = S.CustomerID
INNER JOIN Salespeople AS P ON P.Username = S.Username
WHERE S.SalesPrice < V.InvoicePrice
ORDER BY S.SaleDate DESC, PriceRatio DESC
AS MainTable3;
```

- If price ratio less than or equal to 95%:
  - Highlight background in red

```
--Get highlighted rows
SELECT * FROM MainTable3
WHERE PriceRatio <= 0.95;
```

- Push ***Close*** to return to **Main Menu**

View Average Time in Inventory Report

- Page open by pushing ***View Average Time in Inventory Report*** button on **Main Menu**
- Get the sold date in Sales Events

- Find Vehicles associated with Sales Events; Get the date added and type of Vehicles
  - Get the inventory time of each sold Vehicle by sold date - date added
  - Average the inventory time by type, as type average inventory time
- Display type, type average inventory time in rows
- If no sales history:
  - Display type: “N/A”

```
With AllTypes AS (
	SELECT 'Car' AS Type UNION SELECT 'SUV' UNION SELECT 'Truck'
	UNION SELECT 'Convertible' UNION SELECT 'Van MiniVan'
),
TypeInfo AS (
	SELECT Vin, 'Car' AS Type FROM Cars
	UNION SELECT Vin, 'SUV' AS Type FROM SUVs
	UNION SELECT Vin, 'Van MiniVan' AS Type FROM VanMiniVans
	UNION SELECT Vin, 'Truck' AS Type FROM Trucks
	UNION SELECT Vin, 'Convertible' AS Type FROM Convertibles
)
SELECT AllTypes.Type, IFNULL(A.AveageTime, 'N/A') AS AverageTimeInInventory
FROM AllTypes
LEFT OUTER JOIN
	(SELECT T.Type, AVG(DateAdded - S.SaleDate+1) AS AverageTime
	FROM Vehicles
	INNER JOIN SalesEvents AS S ON A.Vin = S.Vin
	INNER JOIN TypeInfo AS T ON T.Vin =Vehicles.Vin
	GROUP BY Type) AS A
ON AllTypes.Type = A.Type
ORDER BY Type ASC;
```

- Push ***Close*** to return to **Main Menu**

View Part Statistics Report

- Page open by pushing ***Part Statistics Report*** button on **Main Menu**
- Get the part price, vendor name in Parts
- Find the Needs for the Parts; Get the quantity used in Needs
  - Aggregate quantity used of each Part as part total quantity
  - Get the total cost of each part by part price * part total quantity
  - Sum the part total quantity by vendor name, as vendor total part quantity
  - Sum the total cost of each part by part by vendor name, as total cost of vendor
- Display the vendor names, vendor total part quantity, and total cost of vendor

```
SELECT VendorName,  SUM(QuantityUsed * Price) AS TotalCost,
	SUM(QuantityUsed) AS NumberOfPart
FROM Parts GROUP BY VendorName
ORDER BY TotalCost DESC;
```

- Push ***Close*** to return to **Main Menu**

View Monthly Sales Report

- Page open by pushing ***Monthly Sales Report*** button on **Main Menu**

(Subtask: **view year and month sales**)

- Get the sale date, sale price in Sales Events
- Find the Vehicle associated with Sales Events; get the invoice price in Vehicle
  - Aggregate number of Sales Events by month and by year, as month/year sold vehicle count
  - Sum the invoice price by month/year as month/year invoice price
  - Sum the sale price by month/year as month/year sale income
  - Get the month/year net income by month/year sale price – month/year invoice price
  - Get the month/year ratio by month/year sale price / month/year invoice price
- If no sales in year/month:
  - Exclude the year/month
- Sort the result by year and month descending
- Display year, month, month/year sold vehicle count, month/year sale income, month/year net income or month/year ratio

```
SELECT  Date_FORMAT(SaleDate, '%Y-%m') AS SaleYearMonth, COUNT(*) AS SaleCount,
        SUM(S.SalePrice) AS SaleIncome, SUM(S.SalePrice – V.InvoicePrice) AS SaleNetIncome,
        SUM(S.SalePrice) / SUM(V.InvoicePrice) AS SaleRatio
FROM SalesEvents AS S
INNER JOIN Vehicles AS V
ON V.Vin = S.Vin
GROUP BY SaleYearMonth
ORDER BY SaleYearMonth DESC
AS MainTable4;
```

- If month/year ratio greater than or equal to 125%:

```
--Get highlighted to green
SELECT * FROM MainTable4
WHERE SaleRatio >= 1.25;
```

  - Highlight row in green background
- If month/ratio less than or equal to 110%:

```
--Get highlighted to yellow
SELECT * FROM MainTable4
WHERE SaleRatio <= 1.1;
```

  - Highlight row in yellow background

(Subtask: **View salespeople performance**)

- Push the ***More Detail*** button of each month/year to open drill down
- Find the Salespersons of Sales Events; get the name of Salespersons
  - Aggregate the number of sales of each Salesperson by month/year as salesperson month/year sales count
  - Sum the sale price of Salespersons income by salesperson month/year income

- Sort the salesperson month/year sales count descending and by salesperson month/year income descending
- Display the Salesperson’s name, salesperson month/year sales count, salesperson month/year income
- Push ***Cancel*** to close the drill-down

```
--if click on a year row, assume the SaleDate of the row is '$SaleYearMonth'
SELECT CONCAT(P.FirstName, ' ',P.LastName) AS SalespersonName, COUNT(*) AS SaleCount,
        SUM(S.SalePrice) AS SalespersonIncome
FROM SalesEvents AS S
INNER JOIN Salespeople AS P
ON P.Username = S.Username
WHERE YEAR( S.SaleDate) = YEAR('$SaleYearMonth')
        AND MONTH( S.SaleYearMonth) = MONTH('$SaleYearMonth')
GROUP BY P.Username
ORDER BY YearSaleCount DESC, SalespersonIncome DESC;
```

- Push ***Close*** to return to **Main Menu**