Jaunty Jalopies Database Design

Table of Content:

Table of Content: 1
Assumptions and Requirements 2
View Customer Report 11
Search Vehicle 23
Add Vehicle Sale 28
View Sales Report 4
Add/Update Repair 29
View Repair Report 16
View Vehicle Report 19
Login 2
Add Vehicle 22
View Vehicle Detail 25
Add/Update manufacturer 33

Assumptions and Requirements

mySQL 8.0 and above.

Login

Abstract Code

- Logged-in User enters *username* (‘$Username’), *password* (‘$Password’) input fields in **Login** form.
- Validate for both *username* and *password* input fields:
    - When ***Login*** button is clicked:

```
SELECT userName, password
FROM LoggedInUser
WHERE userName = '$userName' ANDpassword = '$password'
```

    - If record does not exist, Go back to **Login** form, throw error message “Wrong password!”.
    - Else: calculate the user role_type
    - Union all user tables, get their userNames and assign respective role_type for each set of userNames from the following Tables: **Salesperson**, **ServiceWriter**, **InventoryClerk**, **Manager** and **Owner**.

```
WITH
        all_users AS(SELECT userName, "Salesperson" AS role_type
        FROM Salesperson
        UNION
        SELECT userName, "ServiceWriter" AS role_type
        FROM ServiceWriter
        UNION
        SELECT userName, "InventoryClerk" AS role_type
        FROM InventoryClerk
        UNION
        SELECT userName, "Manager" AS role_type
        FROM Manager
        UNION
        SELECT userName, "Owner" AS role_type
        FROM Owner)

SELECT role_type
FROM all_users
WHERE userName '$userName'
```

```
SELECT LoggedInUser.userName, LoggedInUser.password
FROM Salesperson
JOIN LoggedInUser
ON Salesperson.userName = LoggedInUser.userName
```

```
WHERE LoggedInUser.userName = '$userName' AND
LoggedInUser.password = '$password'

SELECT LoggedInUser.userName, LoggedInUser.password
FROM ServiceWriter
JOIN LoggedInUser
ON ServiceWriter.userName = LoggedInUser.userName
WHERE LoggedInUser.userName = '$userName' AND
LoggedInUser.password = '$password'

SELECT LoggedInUser.userName, LoggedInUser.password
FROM InventoryClerk
JOIN LoggedInUser
ON InventoryClerk.userName = LoggedInUser.userName
WHERE LoggedInUser.userName = '$userName' AND
LoggedInUser.password = '$password'

SELECT LoggedInUser.userName, LoggedInUser.password
FROM Manger
JOIN LoggedInUser
ON Manger.userName = LoggedInUser.userName
WHERE LoggedInUser.userName = '$userName' AND
LoggedInUser.password = '$password'

SELECT LoggedInUser.userName, LoggedInUser.password
FROM Owner
JOIN LoggedInUser
ON Owner.userName = LoggedInUser.userName
WHERE LoggedInUser.userName = '$userName' AND
LoggedInUser.password = '$password'
```

- If Logged-in User is Inventory Clerk
  - Show, ***Search Vehicle by VIN***, ***Add/Update manufacturer***, ***Add Vehicle*** and ***Log out*** tabs.
  - Upon:
    - Click ***Search Vehicle by VIN*** button – Jump to the **Search Vehicle** task.
    - Click ***Add Vehicle*** button – Jump to the **Add Vehicle** task.
    - Click ***Add/Update manufacturer*** button – Jump to the **Add/Update manufacturer** task.
    - Click ***Log out*** button – Invalidate login session and go back to the **Login** form.
- If Logged-in User is Saleperson
  - Show ***Search Vehicle by VIN***, ***Add/Update Sale*** and ***Log out*** tabs.
  - Upon:

        - Click ***Search Vehicle by VIN*** button – Jump to the **Search Vehicle** task.
        - Click ***Add/Update Sale*** button – Jump to the **Add/Update Sale** task.
        - Click ***Log out*** button – Invalidate login session and go back to the **Login** form.
    - If Logged-in User is Service Writer
        - Show, ***Search Vehicle by VIN***, ***Add/Update Repair Form*** and ***Log out*** tabs.
        - Upon:
            - Click ***Search Vehicle by VIN*** button – Jump to the **Search Vehicle** task.
            - Click ***View Repair Form*** button – Jump to the **Add/Update Repair Form** task.
            - Click ***Log out*** button – Invalidate login session and go back to the **Login** form.
    - If Logged-in User is Manager
        - Show ***Search Vehicle by VIN***, ***View Report*** and ***Log out*** tabs.
        - Upon:
            - Click ***Search Vehicle by VIN*** button – Jump to the **Search Vehicle** task.
            - Click ***View Report*** button – Jump to the **View Report** task.
            - Click ***Log out*** button – Invalidate login session and go back to the **Login** form.
    - If Logged-in User is Roland Around/Owner
        - Show, ***Search Vehicle by VIN***, ***View Report***, ***Add Vehicle***, ***Add/Update Sale*** and ***Log out*** tabs.
        - Upon:
            - Click ***Search Vehicle by VIN*** button – Jump to the **Search Vehicle** task.
            - Click ***Add Vehicle*** button – Jump to the **Add Vehicle** task.
            - Click ***Add/Update Sale*** button – Jump to the **Add/Update Sale** task.
            - Click ***View Report*** button – Jump to the **View Report** task.
            - Click ***Log out*** button – Invalidate login session and go back to the **Login** form.
- Else *username* and *password* input fields are both invalid, display **Login** form, with error message “no such user”.

View Sales Report

Abstract Code

- The Managers and Roland Around/Owner click ***View Report*** button.
    - If access to ***Sales by Color Report***.
        - Get Distinct Colors in **Color** table.
        - Calculate the last available sale date by getting the max purchase date from **SalesTransaction** table.
        - Calculate the date of 30 days before last available sale.
        - Calculate the date of 1 year before last available sale.
        - Calculate a set of VIN that are of single color.
        - Calculate a set of VIN that are of multiple color.
        - Calculate a set of color options including "multiple" and individual colors.
        - For Sale Transaction Purchase Date <= 30 days
            - Inner join **Vehicle** table with **SalesTransaction** table with condition that:
                - sales date in **SalesTransaction** within 30 days from last available sale.
                - VIN in **SalesTransaction** Table = VIN in **Vehicle**
            - Count number of vehicles grouping by Color. Display 0 if the count is 0, for that color option.
        - Sale Transaction Purchase Date < 1 year
            - Inner join **Vehicle** table with **SalesTransaction** table with condition that:
                - Purchase Date in **Sale Transaction Purchase Date** within 1 year from last available sale.
                - VIN in **Sales** = VIN in **Vehicle**
            - Count number of vehicles grouping by Color. Display 0 if the count is 0, for that color option.
        - Sales of overall
            - Inner join **Vehicle** table with **SalesTransaction**table with condition that:
                - VIN in **Sale Transaction** = VIN in **Vehicle**
            - Count number of vehicles grouping by Color. Display 0 if the count is 0, for that color option.
        - **Display the count for each color option, in 3 different time ranges.**
    -

```
SET sql_mode =
'STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_
DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
SELECT @last_avaliable_sale_date := MAX(purchase_date) FROM
SalesTransaction;
SELECT @month_cutoff :=
DATE_ADD(@last_avaliable_sale_date,INTERVAL -30 DAY);
SELECT @year_cutoff := DATE_ADD(@last_avaliable_sale_date,INTERVAL -
1 YEAR);

WITH
```

```
single_color_vehicle AS (
SELECT V.VIN, VC.color AS color
  FROM Vehicle_Color V NATURAL JOIN Vehicle_Color VC
  GROUP BY V.VIN
  HAVING COUNT(VC.VIN) = 1
),

multi_color_vehicle AS (
  SELECT V.VIN, VC.color AS color
  FROM Vehicle_Color V NATURAL JOIN Vehicle_Color VC
  GROUP BY V.VIN
  HAVING NOT COUNT(V.VIN) = 1
),

vehicles_by_color AS (
  SELECT MV.VIN, 'multiple' as color
  FROM multi_color_vehicle MV
  UNION
  SELECT SV.VIN, SV.color
  FROM single_color_vehicle SV
),

one_month_report AS (
  SELECT V.color, IFNULL(COUNT(V.VIN), 0) AS one_month_sales_count
  FROM vehicles_by_color V NATURAL JOIN SalesTransaction S
  WHERE S.purchase_date >= @month_cutoff
  GROUP BY V.color
),

one_year_report AS (
  SELECT V.color, COUNT(V.VIN) AS one_year_sales_count
  FROM vehicles_by_color V NATURAL JOIN SalesTransaction S
  WHERE S.purchase_date >= @year_cutoff
  GROUP BY V.color
),

all_time_report AS (
  SELECT V.color, COUNT(V.VIN) AS alltime_sales_count
  FROM vehicles_by_color V NATURAL JOIN SalesTransaction S
  GROUP BY V.color
),

report AS (
  SELECT all_time_report.color AS Color,
IFNULL(one_month_report.one_month_sales_count, 0) AS
one_month_sales_count, IFNULL(one_year_report.one_year_sales_count,
```

```
0) AS one_year_sales_count,
IFNULL(all_time_report.alltime_sales_count,0) AS alltime_sales_count
  FROM one_month_report
  RIGHT JOIN one_year_report
  ON one_month_report.color = one_year_report.color
  RIGHT JOIN all_time_report
  ON one_year_report.color = all_time_report.color
),

all_color_options AS(
  SELECT Color
  FROM report
  UNION
  SELECT color
  FROM Color
  )

SELECT Color, one_month_sales_count, one_year_sales_count ,
alltime_sales_count
FROM report R
NATURAL JOIN all_color_options
ORDER BY Color ASC
```

- If access to ***Sales by Type Report***:
  - Manually from a set of Distinct types.
  - Calculate the last available sale date by getting the max purchase date from **SalesTransaction** table.
  - Calculate the date of 30 days before last available sale.
  - Calculate the date of 1 year before last available sale.
  - For Sales date <= 30 days
    - Inner join **Vehicle** table with **SalesTransactions**table with condition that:
      - sales date in **Sales Form** within 30 days from last available sale.
      - VIN in **SalesTransactions** = VIN in **Vehicle**
    - Count number of vehicles grouping by vehicle_type. Display 0 if the count is 0.
  - For Sales date 1 year
    - Inner join **Vehicle** table with **SalesTransactions** table with condition that:
      - sales date in **SalesTransactions** within 1 year 365 days from last available sale.
      - VIN in **SalesTransactions**= VIN in **Vehicle**
    - Count number of vehicles grouping by vehicle_type Display 0 if the count is 0.
  - Sales over all

- Inner join **Vehicle** table with **SalesTransactions** table with condition that:
  - VIN in **SalesTransactions** = VIN in **Vehicle** and
- Count number of vehicles grouping by vehicle_type Display 0 if the count is 0.

- **Sort by vehicle type**
- **Display the count for each type, in 3 different time ranges.**

```
SET sql_mode =
'STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_
DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
SELECT @last_avaliable_sale_date := MAX(purchase_date) FROM
SalesTransaction;
SELECT @month_cutoff :=
DATE_ADD(@last_avaliable_sale_date,INTERVAL -30 DAY);
SELECT @year_cutoff := DATE_ADD(@last_avaliable_sale_date,INTERVAL -
1 YEAR);

WITH
vehicle_type AS(
 SELECT 'Car' AS vehicle_type
 UNION
 SELECT 'Convertible' AS vehicle_type
 UNION
 SELECT 'Truck' AS vehicle_type
 UNION
 SELECT 'VAN_MiniVAN' AS vehicle_type
 UNION
 SELECT 'SUV' AS vehicle_type
 ),

one_month_report AS (
 SELECT V.vehicle_type, COUNT(V.VIN) AS one_month_sales_count
 FROM SalesTransaction S NATURAL JOIN Vehicle V
 WHERE S.purchase_date >= @month_cutoff
 GROUP BY vehicle_type
),
one_year_report AS (
 SELECT V.vehicle_type, COUNT(V.VIN) AS one_year_sales_count
 FROM SalesTransaction S NATURAL JOIN Vehicle V
 WHERE S.purchase_date >= @year_cutoff
 GROUP BY vehicle_type
),

all_time_report AS (
 SELECT V.vehicle_type, COUNT(V.VIN) AS alltime_sales_count
 FROM SalesTransaction S NATURAL JOIN Vehicle V
```

```
  GROUP BY vehicle_type
),

report AS(
  SELECT all_time_report.vehicle_type AS vehicle_type,
one_month_report.one_month_sales_count,
one_year_report.one_year_sales_count,
all_time_report.alltime_sales_count
  FROM one_month_report
  RIGHT JOIN one_year_report
  ON one_month_report.vehicle_type = one_year_report.vehicle_type
  RIGHT JOIN all_time_report
  ON one_year_report.vehicle_type = all_time_report.vehicle_type
  )

SELECT VT.vehicle_type, IFNULL(R.one_month_sales_count, 0) AS
one_month_sales_count, IFNULL(R.one_year_sales_count, 0) AS
one_year_sales_count, IFNULL(R.alltime_sales_count,0) AS
alltime_sales_count
FROM report R
RIGHT JOIN vehicle_type VT
ON R.vehicle_type = VT.vehicle_type
ORDER BY R.vehicle_type ASC
```

- If access to ***Sales by manufacturer Report***:

```
SET sql_mode =
'STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_
DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
SELECT @last_avaliable_sale_date := MAX(purchase_date) FROM
SalesTransaction;
SELECT @month_cutoff :=
DATE_ADD(@last_avaliable_sale_date,INTERVAL -30 DAY);
SELECT @year_cutoff := DATE_ADD(@last_avaliable_sale_date,INTERVAL -
1 YEAR);

WITH
one_month_report AS (
  SELECT V.manufacturer, COUNT(V.VIN) AS one_month_sales_count
  FROM SalesTransaction S NATURAL JOIN Vehicle V
  WHERE S.purchase_date >= @month_cutoff
  GROUP BY manufacturer
),
one_year_report AS (
   SELECT V.manufacturer, COUNT(V.VIN) AS one_year_sales_count
  FROM SalesTransaction S NATURAL JOIN Vehicle V
  WHERE S.purchase_date >= @year_cutoff
```

```
  GROUP BY manufacturer
),

all_time_report AS (
   SELECT V.manufacturer, COUNT(V.VIN) AS alltime_sales_count
  FROM SalesTransaction S NATURAL JOIN Vehicle V
  GROUP BY manufacturer
),

report AS(
  SELECT all_time_report.manufacturer AS manufacturer,
one_month_report.one_month_sales_count,
one_year_report.one_year_sales_count,
all_time_report.alltime_sales_count
  FROM one_month_report
  RIGHT JOIN one_year_report
  ON one_month_report.manufacturer = one_year_report.manufacturer
  RIGHT JOIN all_time_report
  ON one_year_report.manufacturer = all_time_report.manufacturer
  )

SELECT R.manufacturer, IFNULL(R.one_month_sales_count, 0) AS
one_month_sales_count, IFNULL(R.one_year_sales_count, 0) AS
one_year_sales_count, IFNULL(R.alltime_sales_count,0) AS
alltime_sales_count
FROM report R
ORDER BY R.manufacturer AS
```

- If access to ***Monthly Sales Report***:
  - **Load Summary Page**
  - Fetch all data from **SalesTransaction** Table and natural join with Vehicle table using VIN.
  - Count the number of items, save as a variable total_number_sold. Do this for each month and for each year.
  - Sum up the Sold Price for Sale Transactions, save as a variable income Do this for each month and for each year.
  - Find the Vehicle using VIN (from SalesTransaction), and get Invoice Price. Sum up Invoice price for each vehicle sold during that time period, save as total_invoice. Do this for each month and for each year.
  - Arrange the above data in each year and month. For each time period, calculate income by using income – total_invoice. Also calculate income / total_invoice for each month and year, as ratio.
  - Arrange and order by year and month descending and Display the above data by year and month.

```
SELECT YEAR(S.purchase_date) AS Year, MONTH(S.purchase_date) AS
Month, COUNT(S.VIN) AS total_number_sold, SUM(S.sold_price) AS
income, (SUM(S.sold_price) - SUM(V.invoice_price)) AS net_income,
CONCAT((SUM(S.sold_price) / SUM(V.invoice_price)*100),'%') AS ratio
FROM SalesTransaction S NATURAL JOIN Vehicle V
GROUP BY YEAR(S.purchase_date), MONTH(S.purchase_date)
ORDER BY YEAR(S.purchase_date) DESC, MONTH(S.purchase_date)
DESC
```

- If the ratio is >= 125%, highlight with green background.
- If the ratio is >= 110%, highlight with yellow background.
- Show ***Top Sales Report*** Button for each year and month
- **When the *Top Sales Report* in pushed for a *year*** '$Year' ***Month*** '$Month'**, Load the sales data for selected year and month. For each time priod, do the following:**
    - Natural join the **SalesTransactions** table, **Vehicle** table and **SalesPerson** in this order.
    - Group by SalesPerson username.
    - Count the number of items and sum up the sold_price under each Group (by SalePerson).
    - Sort the group first by number of items, and then by total sold_price, and finally by last_name and first_name.
    - Display the SalePerson's First Name and Last Name, alongside with the calculated number and total_sales for that year and month.

```
SELECT SP.first_name, SP.last_name, COUNT(S.VIN) AS
total_number_sold, SUM(S.sold_price) AS total_sales
FROM SalesTransaction S NATURAL JOIN Vehicle V NATURAL
JOIN Salesperson SP
WHERE YEAR(S.purchase_date) = '$Year' AND
MONTH(S.purchase_date) = '$Month'
GROUP BY S.userName
ORDER BY total_number_sold DESC, total_sales DESC,
last_name, first_name
```

View Gross Customer Income Report

Abstract Code

- The Managers and Roland Around/Owner click ***View Report*** button.
    - Display Gross Customer Income report:
        - **Load the top 15 Customers**, the gross income received from them via Vehicle sales and/or Repairs (including any Repairs in progress)
            - Fetch all Sale Transactions

- Group the SalesTransaction by Customer's identifier (either Diver License Number or Tax Identification Number)
- Sum up the total Sold Price for each customer. Save this view has GrossSale.
- Fetch all Repairs
- Group the Repairs by Customer's identifier
- Sum up of the total cost (by adding up labor charge and parts costs for each repair). Save this view as GrossRepair.
- Both views share the same common Customer identifier.
- Combine GrossSale and GrossRepair, summing up the GrossSale value and GrossRepair value for each customer, as GrossCustomerIncome for each customer. Save this view including all customers as GrossCustomer.Sort the GrossCustomer by the GrossCustomerIncome descending, and by last Sale Transaction Purchase Date or Repair Start Date.
- Display the top 15 items from GrossCustomer for the following attributes, for each one of them:
- **Find** and Display the Customer's Name (first/last for individuals or business name for business)
- **Find the** Gross Customer date of the first Sale Transaction Purchase Date and Repair Start date. Display the earlier one.
- **Find** the most recent Sale Transaction Date and Repair Start Date. Display the most recent one.
- Count the number of sales, the number of repairs, and the GrossCustomerIncome

- Left out join Show "More Detail" Button near each of the customer Name

```
WITH  Repair_Form as
(
SELECT    r.start_date
          , r.VIN
          , r.customerID
          , r.completion_date
          , r.odometer_readout
          , r.userName
          , sum(r.labor_charge) as labor_cost
          , sum(p.quantity * p.price) as part_cost
          , count(r.start_date) as number_of_repair
FROM Repair r
LEFT OUTER JOIN Part p
ON p.VIN = r.VIN and p.start_date = r.start_date and p.customerID
= r.customerID
GROUP BY r.customerID
),

 Vehicle_from_sale as
(
```

```
SELECT   v.VIN
         , v.model_year
         , v.model_name
         , v.manufacturer
         , v.invoice_price
         , s.sold_price
         , s.purchase_date,
         , count(s.VIN) as Number_of_Sales
From Vehicle v
Left outer join SalesTransaction s on v.VIN = s.VIN and
v.customerID = s.customerID
Group by s.customerID
),

Vehicle_Form as
(
Select   vs.VIN
         , vs.purchase_date
         , vs.number_of_Sales
         , vs.sold_price
         , vs.model_year
         , vs.model_name
         , vs.manufacturer
         , vs.invoice_price
         , rf. start_date
         , rf. number_of_repair
         , rf. completion_date
         , rf. odometer_readout
         , rf.part_cost
         , rf.labor_cost
         , rf.userName
         , sum(rf.part_cost + rf.labor_cost) as total_cost
From Vehicle_from_sale vs
Left outer join Repair_Form rf on vs.VIN = rf.VIN and vs.customerID
= rf.customerID
Group by rf.customerID , rf.start_date
),

 CustomerI as
(
Select    c.customerID
         , c.email
         , c.phone_number
         , c.street_address
         , c.city
         , c.state
         , c.postal_code
```

```
        , i.Driver_License_Number
        , concat(i.first_name, i.last_name) as Name
From Customer c
Full outer join Individual i on i.Driver_License_Number =
c.customerID
),


 CustomerB as
(
Select   c.customerID
        , c.email
        , c.phone_number
        , c.street_address
        , c.city
        , c.state
        , c.postal_code
        , b.Tax_Identifiation_Number
        , b.business_name AS Name
        , b.primary_contact_name
        , b.primary_contact_title
From Customer c
Full outer join Business b on b.Tax_Identification_Number =
c.customerID
),

 CustomerU as
(
Select   ci.customerID
        , ci.Name
        , " " as [Primary Contact Name]
        , " " as [Primary Contact Title]
From CustomerI ci
UNION
Select  cb.customerID
        , cb.Name
        , cb.Primary_Contact_Name as [Primary Contact Name]
        , cb.Primary_Contact_Title as [Primary Contact Title]
From CustomerB cb
);

Select  cu.Name,
        , min(vf.purchase_date) as [First Sale Date]
        , vf.start_date as [Repair Start Date]
        , max(vf.purchase_date) as [Most Recent Sale Date]
        , vf.number_of_Sales as [Number of Sales]
        , vf.number_of_repair as [Number of Repair]
```

```
        , TOP 15 sum( vf.total_cost + vf.sold_price) as [Gross
        Income]
From CustomerU cu
Left outer join Vehicle_Form vf on cu.customerID = vf.customerID
Group by cu.customerID
Order by sum( vf.total_cost + vf.sold_price) desc ,
          max( vf.purchase_date) desc ,
          vf.start_date desc;
```

- When *More Detail* Button is pushed**:**
  - **Load Vehicle sales section:**
    - Lookup Vehicle and Sale Transaction using VIN and Customer Identifier.
    - For each Sale Transaction:
    - Display Sale Date, Sold Price, VIN, year, Lookup manufacturer and display manufacturer Name, Model and Lookup Logged-in User and display Salesperson Name.
    - Sort Vehicle SalesTransactions by sale date descending and VIN ascending.

```
Select cu.Name
        , vf.purchase_date as [Sale Date]
        , vf.sold_price as [Sold Price]
        , vf.VIN
        , vf.manufacturer
        , vf.model_year as [Model Year]
        , vf.model_name as [Model Name]
        , vf.Salesperson_Name as [Salesperson
        Name]
From CustomerU cu
Left outer join Vehicle_Form vf on cu.customerID =
vf.customerID
Where vf.purchase_date is not null
Order by vf.purchase_date desc ,
          vf.VIN desc;
```

  - **Load repair section:**
    - Search for Repairs using VIN and Customer IdentifieR.
    - For each Repair, display Start date, End date (if the Repair is not finished, this should not display any value). Lookup Vehicle using VIN. Display the VIN of the repaired Vehicle, the Odometer Reading, Parts Cost,

Labor Cost, Total Cost, and Lookup the Logged-in User (the Service Writer) who opened the Repair and display First Name and Last Name

- Sort Repairs by Start Date descending, End Date descending

```
Select   cu.Name
         , vf.start_date as [Start Date]
         , vf.completion_date as [End Date]
         , vf.VIN
         , vf.odometer_readout as [Odometer Reading]
         , vf.part_cost as [Parts Cost]
         , vf.labor_cost as [Labor Cost]
         , vf.total_cost as [Total Cost]
         , r.userName as [User Name]
From CustomerU cu
Full outer join Repairs r on r.customerID = cu.customerID
Left outer join Vehicle_Form vf on cu.customerID =
vf.customerID
                  Where vf.start_date is not null
                  Order by vf.start_date desc ,
                           vf.completion_date is null ,
                           vf.completion_date desc ,
                           vf.VIN asc;
```

View Repair Report

- The Managers and Roland Around/Owner click ***View Report*** button.
  - If they want to see ***Repairs by manufacturer/Type/Model Reports***:
    - Fetch all Repairs.
    - Each Repair is tied with a Vehicle, group the Repairs by Vehicle manufacturer Name, regardless of whether Repair Completion Date is null or not.
    - For each manufacturer, do the following:
      - Count the number of Repairs;
      - Adding up all the Parts Cost for all of the Repairs for this manufacturer.
      - Adding up all the Labor Cost for all of the Repairs for this manufacturer.

- Adding up the above two results to get the Total Repair Costs for this manufacturer.
- Load all the manufacturer Names. For manufacturer Name not present in the result, add them to the result with Number of Repair 0 and Cost 0.
- Display the calculated information above for each manufacturer in manufacturer name ascending.

```
WITH repair_records AS (
SELECT R.VIN AS VIN, R.start_date AS start_date, labor_cost,
part_cost, COALESCE(labor_cost + part_cost, labor_cost, part_cost,
0) AS total_cost
FROM (
(SELECT VIN, start_date, labor_charge AS labor_cost FROM Repair)
AS R
LEFT JOIN
(SELECT VIN, start_date, sum(price * quantity) AS part_cost FROM
Part GROUP BY VIN, start_date) AS P
ON R.VIN = P.VIN AND R.start_date = P.start_date)
)

SELECT V.manufacturer as manufacturer, COUNT( DISTINCT R.VIN,
R.start_date) AS repair_count, SUM(R.labor_cost) AS
total_labor_cost, SUM(R.total_cost) AS total_repair_cost,
SUM(R.part_cost) AS total_part_cost
FROM
(Vehicle AS V
LEFT JOIN
repair_records AS R
ON V.VIN = R.VIN)
GROUP BY V.manufacturer
ORDER BY V.manufacturer ASC;
```

- For every manufacturer that have Repairs: User can click on ***More Detail*** button of selected $manufacturer to **load <u>drill-down report</u>** (sorted by repair count descending by Vehicle Type first then models)
    - Search for Repairs filtering only for this $manufacturer, Group by Vehicle Type.
    - For each Vehicle Type that have Repairs, do the following:
        - Count the number of Repairs and Display it.
        - Summing up all the total Parts Cost and display It.
        - Summing up all the Labor Cost and display it.
        - Adding up the above two results, save as total Total Cost and display it.
        - Sorted by Repair Count descending.

```
WITH repair_records AS (
```

```
SELECT R.VIN AS VIN, R.start_date AS start_date, labor_cost,
part_cost, COALESCE(labor_cost + part_cost, labor_cost,
part_cost, 0) AS total_cost
FROM (
(SELECT VIN, start_date, labor_charge AS labor_cost FROM
Repair) AS R
LEFT JOIN
(SELECT VIN, start_date, SUM(price * quantity) AS part_cost
FROM Part GROUP BY VIN, start_date) AS P  ON R.VIN = P.VIN
AND R.start_date = P.start_date)
)

SELECT V.vehicle_type AS vehicle_type, COUNT(DISTINCT
R.VIN, R.start_date) AS repair_count, SUM(R.labor_cost) AS
total_labor_cost, SUM(R.total_cost) AS total_repair_cost,
SUM(R.part_cost) AS total_part_cost
FROM
( repair_records AS R
INNER JOIN
Vehicle AS V
ON R.VIN = V.VIN AND V.manufacturer = '$manufacturer')
GROUP BY V.vehicle_type
ORDER BY V.vehicle_type ASC, repair_count DESC;
```

- **Group the Repairs under this Vehicle Type by Model.**
- For each model that have repairs in that Vehicle Type:
  - Summing up all the total Parts Cost and display It.
  - Summing up all the Labor Cost and display it.
  - Adding up the above two results, save as total Total Cost and display it.
  - Sorted by Repair Count descending.

```
WITH repair_records AS (
SELECT R.VIN AS VIN, R.start_date AS start_date,
labor_cost, part_cost, COALESCE(labor_cost +
part_cost, labor_cost, part_cost, 0) AS total_cost
FROM (
(SELECT VIN, start_date, labor_charge AS labor_cost
FROM Repair) AS R
LEFT JOIN
(SELECT VIN, start_date, SUM(price * quantity) AS
part_cost FROM Part GROUP BY VIN, start_date) AS
P  ON R.VIN = P.VIN AND R.start_date = P.start_date)
),

vehicle_type_report AS (SELECT V.vehicle_type AS
vehicle_type, COUNT(DISTINCT R.VIN, R.start_date)
```

```
AS repair_count, SUM(R.labor_cost) AS
total_labor_cost, SUM(R.total_cost) AS
total_repair_cost, SUM(R.part_cost) AS
total_part_cost, V.model_name AS model_name
FROM
( repair_records AS R
INNER JOIN
Vehicle AS V
ON R.VIN = V.VIN AND V.manufacturer =
'$manufacturer')
GROUP BY V.vehicle_type, V.model_name
ORDER BY V.vehicle_type ASC, repair_count DESC;
)

-- for each '$vehicle_type' drill down model details:
SELECT model_name, repair_count, total_labor_cost,
total_part_cost , total_repair_cost
FROM vehicle_type_report
WHERE vehicle_type = '$vehicle_type'
GROUP BY model_name
ORDER BY repair_count DESC;
```

- If access to ***Parts Statistics Report***:
  - Fetch all the Parts by using VIN, Start Date and Part Number.
  - Group the Parts by Vendor Name.
  - For each group, do the following:
    - Summing up the Quantity for all parts under this Vendor.
    - Calculate Quantity x Price for each part, sum up for each part.
    - Display the above information, and Vendor Name.

```
SELECT vendor_name, sum(quantity) AS total_supply_quantity, sum(price
* quantity) AS total_spent_on_parts
FROM Part
GROUP BY vendor_name
ORDER BY vendor_name;
```

View Vehicle Report

- The Managers and Roland Around/Owner click ***View Report*** button.
  - If access to ***Average Time in Inventory Reports***:
    - Get a list of all vehicle types.
    - Fetch all the Vehicles from **Vehicle** table and natural join the **SalesTransaction** table.
    - Group by Vehicle Type.
    - For each group (items under each Vehicle Type), do the following:

- Find the SalesTransaction for that Vehicle using VIN. Calculate time_int_inventory as the day differnece between SalesTransaction purchase_date and Vehicle added_date)
- Take the average for all the calculated time_int_inventory for this group.
- **Display Average Time in Inventory information for all groups.**

```
WITH
vehicle_type_options AS(
  SELECT 'Car' AS vehicle_type
  UNION
  SELECT 'Convertible' AS vehicle_type
  UNION
  SELECT 'Truck' AS vehicle_type
  UNION
  SELECT 'VAN_MiniVAN' AS vehicle_type
  UNION
  SELECT 'SUV' AS vehicle_type
  ),

report AS(
SELECT vehicle_type, AVG(DATEDIFF(S.purchase_date,V.added_date) +
1) AS avg_days_in_inventory
FROM Vehicle V NATURAL JOIN SalesTransaction S
GROUP BY vehicle_type
)

SELECT VO.vehicle_type, IFNULL(R.avg_days_in_inventory, 'N/A') AS
avg_days_in_inventory
FROM vehicle_type_options VO LEFT JOIN report R
ON VO.vehicle_type = R.vehicle_type
ORDER BY vehicle_type ASC
```

- If access to ***Below Cost Sales Report***:
  - NATURAL JOIN **SalesTransaction**, **Vehicle** and **SalesPerson**, in this order .
  - Get a set of tuples having this structure (customerId, name) where name is first and last for individuals or business name for business.
  - Calculate Sold Price/Invoice Price as a percentage and display it.
  - **If the ratio >=1.0, skip this and do not display anything.**
  - **Else:**
    - Display the purchase date, the Invoice Price, the Sold Price, and the calculated ratio.
    - **Lookup the customer** using customerId (done at the natural join step), and display the customer's name.

- **Lookup the salesperson** using username (done at the natural join step), display name of the Salesperson for the Sale Transaction.
- If ratio <= 95%, the background of that row should be highlighted red

- Sort the list by sales date descending and ratio descending.

```
WITH
report AS(
  SELECT S.purchase_date, V.invoice_price, S.sold_price,
CONCAT(S.sold_price/V.invoice_price * 100,'%') AS ratio,
SP.first_name, SP.last_name, S.customerID
  FROM SalesTransaction S NATURAL JOIN Vehicle V NATURAL JOIN
Salesperson SP
  WHERE S.sold_price < V.invoice_price
),

customer_i AS(
  SELECT C.customerID, concat(I.first_name, I.last_name) AS Name
  FROM Customer C
  RIGHT JOIN Individual I
  ON I.Driver_License_Number = C.customerID
),

customer_b AS(
  SELECT C.customerID, B.business_name AS Name
  FROM Customer C
  RIGHT JOIN Business B
  ON B.Tax_Identification_Number = C.customerID
),

customer_u AS(
  SELECT CI.customerID, CI.Name
  FROM customer_i CI
  UNION
  SELECT CB.customerID, CB.Name
  FROM customer_b CB
)

SELECT R.purchase_date, R.invoice_price, R.sold_price, R.ratio, C.Name
AS customer_name, R.first_name AS salesperson_first_name,
R.last_name AS salesperson_last_name
FROM customer_u C NATURAL JOIN report R
ORDER BY R.purchase_date DESC, ratio DESC
```

Add Vehicle

Abstract Code

- Users who login as Inventory Clerk and Roland Around/Owner can click on *"Add Vehicle" button* to **add new Vehicles**:
    - **Load <u>new vehicle form</u>**, the form will wait for user input for all relevant fields such as *VIN, Vehicle Type, Model Name, Model Year, Invoice Price, Description (optional), List Price*, and type-specific fields such as *Number of doors* for Car, *Roof Type* and *Back Seat Count* for Convertible, *Cargo Capacity, Cargo Cover Type (optional) and Number of Rear Axis* for Truck, *Driver Side Backdoor* for van&minivan, *Drivertrain Type and Number of Cupholders* for SUV.
        - User inputs ***type*** , '$vehicle_type'
        - When user click ***manufacturer Name*****, Show** a dropdown of list of manufacturer names.

          ```
          SELECT manufacturer FROM manufacturer
          ```
        - When user click ***Color***, **Show** a dropdown of list of Color to add to the Color section. User can select more than one colors.

          ```
          SELECT color_name FROM Color
          ```
        - If the user clicks ***submit button*** to **add new Vehicle**:

          ```
          INSERT INTO Vehicle (VIN, vehicle_type, model_name, model_year,
          invoice_price, add_date, description, manufacturer)
          VALUES ('$vin', '$vehicle_type', '$model_name', '$model_year',
          '$invoice_price', '$added_date', '$description', '$manufacturer')

          INSERT INTO Car (VIN, number_of_doors)
          VALUES ('$vin', '$number_of_doors')

          INSERT INTO Convertible (VIN, roof_type, back_seat_count)
          VALUES ('$vin', '$roof_type', '$back_seat_count')

          INSERT INTO Truck (VIN, cargo_capacity, cargo_conver_type,
          number_of_rear_axles)
          VALUES ('$vin', '$cargo_capacity', '$cargo_conver_type',
          '$number_of_rear_axles')

          INSERT INTO VAN_MiniVan (VIN, driver_side_door)
          VALUES ('$vin', '$driver_side_door')

          INSERT INTO SUV (VIN, number_of_cupholders)
          VALUES ('$vin', '$roof_type', '$number_of_cupholders')
          ```

            - Validate all fields are not null, except for *description* field and *cargo cover type*.
            - Validate *$Model Year* must be <= current year + 1, and it must be a 4-digit integer, otherwise throw error message.
            - Validate *$Invoice Price* must be a Float.

- Capture today's date as AddedDate using CURDATE()
- **Save data to the database**
- **Redirect to <u>View Vehicle detail page</u>** for the added vehicle (Comment: This is a separate task, refer to **<u>View Vehicle Detail Task</u>**.)

▪ If the user clicks ***Cancel:***

- The new vehicle form will be closed.

Search Vehicle

Abstract Code

- Anonymous User and privileged user land on the **<u>Search Page</u>**. **Load** the total number of vehicles available for purchase in the system (is Sold is False).
- For user who are privileged users, show additional ***search by VIN*** button.
- While ***Search Button*** not pushed, stay on current view.
- To gather the search criteria, when a button is clicked, do the following:
- If the user enters specified *VIN* ($VIN):
    - Create a predicate: Vehicle VIN must be the same as $VIN, used for later search
- If the user enters *Keyword* ($keyword):
    - Create a predicate: keyword must be a substring or entire string of the following Vehicle attributes: manufacturer, model year, model name and description.
- If the user enters *List Price* ($List Price):
    - Create a predicate: Vehicle list price must equal to $List Price
- If the user clicks *manufacturer:*
    - **Fetch all manufacturers**, show dropdown for user selection.
    - Wait for user to click one of the options ($manufacturer)
    - Create a predicate: Vehicle manufacturer must equal to $manufacturer
- If the user clicks *model year:*
    - **Fetch all possible model years,** show dropdown for user selection.
    - Wait for user to click one of the options ($ModelYear)
    - Create a predicate: Vehicle Model Year must equal to $ ModelYear
- If the user clicks *color:*
    - **Fetch all possible colors**, show dropdown for user selection.
    - Wait for user to click one of the options ($Color)
    - Create a predicate: Vehicle Color must equal to $ Color
- After gathering the search criteria, When the ***Search Button*** is pushed:
    - For privileged users using lookup vehicle by VIN:
        - **Lookup the vehicle using VIN**, return the Vehicle regardless of Is Sold value.
    - For users who are either Anonymous User or privileged user, **apply all the predicates** created above. Search for the vehicles.
        - If the user is anonymous user, Inventory Clerk, Salesperson:

            - **Search Vehicle** for those whose Vehicle.IsSold is False, with the gathered conditions (predicates).
        - If the user is Service Writer, Manager, Owner:
            - **Search Vehicles** regardless of the value of IsSold, with the gathered conditions (predicates).
    - If no vehicles meet the search criteria:
        - **Display an error message**: "Sorry, it looks like we don't have that in stock!".
- Once fetched all the Vehicles that met the predicates (criteria), On the **Search Result Page**:
    - If the user is anonymous:
        - Display VIN, Vehicle Type, Model Year, Model Name, manufacturer, Color(s), List Price.
    - If the user is Manager/ Owner:
        - Show Additional Filter by *Sold Vehicles*, *Unsold Vehicles*, *All Vehicles*.
        - If User clicks *Sold Vehicles*:
            - Return Vehicles with Is Sold is 'True'
        - If User clicks *Unsold Vehicles*:
            - Return Vehicles with Is Sold is 'False'
        - If User clicks *All Vehicles*:
            - Return all Vehicles satisfy the predicates.
    - If the keyword matches the description, show X mark.
    - Results are sorted by VIN in ascending order.
    - If user click ***sort button*** on any attribute column, **view sorted table,** display the table sorted by the selected column.
    - If user click ***filter button*** on any attribute column, **view filtered table,** display the table filtered by the selected column.
    - If user click on an ***individual vehicle VIN link***, it will **open a Vehicle Detail Page** of the selected vehicle:
        - **Detail Page** will **load vehicle details,** which includes the VIN, Vehicle Type, attributes for that Vehicle Type, Model Year, Model Name, manufacturer, Color(s), List Price, and the Description for the selected result

```
Select   VIN
         , Vehicle_Type
         , manufacturer
         , model_year
         , model_name
         , Color
         , List_Price
         , (Case when Description like '%Keyword%' then "X" End) as
         Checkbox
From Vehicle
Order by VIN asc
```

View Vehicle Detail

Abstract Code

- **Search the Vehicle** using the Vehicle VIN

```
SELECT VIN, vehicle_type, model_year, model_name, manufacturer, description
FROM Vehicle
WHERE Vehicle.VIN = '$vin'
```

- Select **Vehicle**, Click ***View Vehicle Detail Page*** for the selected **Vehicle**
    - Defined a Standard Display Set as VIN, Vehicle Type, attributes for that Vehicle Type, Model Year, Model Name, manufacturer, Color(s), List Price, description.
    - If user is an Anonymous User: **Display** Standard Display Set

```
SELECT Vehicle.VIN, vehicle_type, model_year, model_name,
manufacturer, color, description
FROM Vehicle
JOIN Vehicle_Color
ON Vehicle.VIN = Vehicle_Color.VIN
WHERE Vehicle.VIN = '$vin'
```

    - If user is an Inventory Clerk: **Display** Standard Display Set + Invoice Price.

```
SELECT Vehicle.VIN, vehicle_type, model_year, model_name,
manufacturer, color, invoice_price, description
FROM Vehicle
JOIN Vehicle_Color
ON Vehicle.VIN = Vehicle_Color.VIN
WHERE Vehicle.VIN = '$vin'
```

    - If user is a Manager or Roland Around/Owner: **Display** the Standard Display Set + Invoice Price + Added Date.

```
SELECT Vehicle.VIN, vehicle_type, model_year, model_name,
manufacturer, color, invoice_price, description, addedDate
FROM Vehicle
JOIN Vehicle_Color
ON Vehicle.VIN = Vehicle_Color.VIN
WHERE Vehicle.VIN = '$vin'
```

        - **Find the Inventory Clerk** for this Vehicle, Display First Name and Last Name
        -
    - If the user is sale person or owner, Show additional ***Sell Vehicle*** button, which will be directed to **Add Vehicle Sale Page**.
- If the user is Manager or Roland Around/Owner:
    - **Find the Sale Transaction** for the Vehicle

```
SELECT sold_price, Purchase_date
FROM SalesTransaction
WHERE VIN = '$vin'
```

        - If Sale Transaction cannot be found:

- **Display Empty sale related information**

- If Sale Transaction can be found,
  - Display Sold Price, Purchase Date
  - **Find the Sale Person**, display First Name and Last Name.

```
SELECT LoggedInUser.first_name, LoggedInUser.last_name
FROM LoggedInUser
JOIN (SELECT SalesTransaction.userName
FROM SalesTransaction
JOIN Salesperson
ON SalesTransaction.userName = Salesperson.userName
WHERE VIN = '$vin') AS U
ON LoggedInUser.userName = U.userName
```

  - **Find the customers** for the Sale Transaction
    - If the Customer is an Individual, display FirstName, Last Name, Phone Number, Email, Address (Street Address, city, state, Postal Code)

```
SELECT Individual.first_name,
       Individual.last_name,
       SC.email,
       SC.phone_number,
       SC.street_address,
       SC.city,
       SC.state,
       SC.postal_code
FROM Individual
JOIN (SELECT SalesTransaction.customerID, Customer.email,
Customer.phone_number, Customer.street_address,
Customer.city, Customer.state, Customer.postal_code
FROM SalesTransaction
JOIN Customer
ON SalesTransaction.customerID = Customer.customerID
WHERE VIN = '$vin') AS SC
ON Individual.customerID = SC.customerID
```

    - If the Customer is a Business, display Tax Identification Number, Business Name, Primary Contact Name, Primary Contact Title, Phone Number, Email, Address (Street Address, city, state, Postal Code)

```
SELECT Business.Tax_Identification_Number,
       Business.business_name,
       Business.business_name,
       Business.primary_contact_number,
       Business.primary_contact,
       SC.email,
       SC.phone_number,
       SC.street_address,
       SC.city,
       SC.state,
```

```
        SC.postal_code
FROM Business
JOIN (SELECT SalesTransaction.customerID, Customer.email,
Customer.phone_number, Customer.street_address,
Customer.city, Customer.state, Customer.postal_code
FROM SalesTransaction
JOIN Customer
ON SalesTransaction.customerID = Customer.customerID
WHERE VIN = '$vin') AS SC
ON Business.customerID = SC.customerID
```

- If the user is Manager or Roland Around/Owner, **Find the Repair** for the Vehicle

```
SELECT VIN, start_date
FROM Repair
WHERE VIN = '$vin'
```

  - If Repair cannot be found, **Display empty Repair section**.
  - If Repair can be found
    - **List all Repairs** (already retrieved when we Find the Repair)
    - For each Repair, **Display** Start date, End date, Labor Charges, Part Cost, and Total Cost (by summing all the labor charges and Part costs)

```
SELECT R.*, (R.labor_charge + R.part_cost) AS total_cost
FROM (SELECT Repair.start_date, Repair.completion_date,
Repair.labor_charge, IF(Part.price = NULL OR Part.quantity = NULL, 0,
SUM(Part.price * Part.quantity)) AS part_cost
FROM Repair
LEFT JOIN Part
ON Repair.VIN = Part.VIN AND Repair.start_date = Part.start_date
GROUP BY Repair.VIN, Repair.start_date
WHERE VIN = '$vin' AND start_date = '$start_date') AS R
```

    - **Find the Service Writer** for each Repair

```
SELECT LoggedInUser.first_name, LoggedInUser.last_name
FROM LoggedInUser
JOIN (SELECT ServiceWriter.userName
FROM Repair
JOIN ServiceWriter
ON Repair.userName = ServiceWriter.userName
WHERE VIN = '$vin' AND start_date = '$start_date') AS S
ON LoggedInUser.userName = S.userName
```

      Display First Name and Last Name
    - **Find the Customer** for each Repair
      - If the Customer is an Individual

        Display First Name and Last Name

```
SELECT Individual.first_name, Individual.last_name
FROM Repair
JOIN Individual
```

```
ON Repair.customerID = Individual.customerID
WHERE VIN = '$vin' AND start_date = '$start_date'
```

- If the Customer is a Business
  Display Business Name

```
SELECT Business.business_name
FROM Repair
JOIN Business
ON Repair.customerID = Business.customerID
WHERE VIN = '$vin' AND start_date = '$start_date'
```

Add Vehicle Sale

- Users who login as Salesperson and Roland Around/Owner clicked ***search vehicle*** button to search vehicle by VIN.
- Then load the **View Vehicle Detail form**, see view vehicle Detail task.
- Click the ***Sell Vehicle*** button then:
  - **Load the Sale Order Form**
    - Click ***Search Customer*** button to search customer
      - Users enters driver's license ('$driver's license'), or tax ID ('$tax ID') input fields.
        - Lookup the customer.
          If the user enters driver_license_number

```
SELECT  first_name, last_name, Driver_License_Number FROM `Individual` WHERE Driver_License_Number= '$driver's license'
```

          If the user enters tax_id

```
SELECT  tax_identification_number, business_name, primary_contact_name, primary_contact_title FROM `Business` WHERE Tax_Identification_Number= '$tax ID'
```

      - If customer is found, load **customer information**:
        - If the customer is an individual, display fetched first_name, last_name, Driver_License_Number.
        - If the customer is a business, display fetched tax_identification_number, Business_Name, primary_contact_name, primary_contact_title
      - Else (If the customer is not found)
        - Show ***Add Customer*** button to add the customer.
        - If ***Add Customer*** is Clicked:
          - Wait for user to input the following fields: *$Phone $Number, $city, $Street Address, $Postal Code, $state, $email(optional)*.
          - Show a dropdown to select $type either Individual or Business

- If $type is Individual:
    - Wait for user to input the following: *first_name* ($`first_name`), last_name ($'last_name'), *friver_license_number* ($`Driver_License_Number`)
    - ***Save*** Button to run the following

```
INSERT INTO ` Individual ` (first_name, last_name, driver_license_number) VALUES ('$first_name', '$last_name', '$Driver_License_Number')
```

- If $type is Business:
    - Wait for user to input the following: *tax_identification_number*($'Tax_Identification_Number'), *business_name* ($'Business_Name'). *primary_contact_name*($'Primary_Contact_Name'), *primary_contact_title* ($'Primary_Contact_Title')
    - ***Save*** Button to run the following:

```
INSERT INTO ` Individual ` (first_name, last_name, Driver_License_Number) VALUES ('$first_name', '$last_name', '$Driver_License_Number')
```

- Entering the *sold_price* ($sold_price).
- Entering the *sold_date* ($purchase_date).
- If logged in as Salesperson AND $sold_price <= 0.95 * invoice_price from the Vehicle (already retrieved from view vehicle Detail task)
    - The system will reject the sale and show an error message.
- Else logged in as Roland Around/Owner OR $sold_price > 95% of invoice_price
    - The system will accept the sale and save the data to database by running the following:
      VIN = $VIN (already retrieved from view vehicle Detail task)

```
INSERT INTO ` SaleTransaction ` (VIN, purchase_date, sold_price) VALUES ('$VIN', CURDATE(), '$ sold_price')
```

- When the ***Cancel*** button is clicked, the add sales window will close.

Add/Update Repair

Abstract Code

- Users who login as Service Writer or Roland Around/Owner click on *"Repair Form"* button to **load the Repair Form**.
- Users enter a *VIN* to **search for Vehicle** on the Repair Form:
    - **Search for Vehicle** using VIN

- If the Vehicle has not been sold (i.e., Is Sold is false)
  - **Display an error message "**This Vehicle is not sold out, should not have any repair".
- If VIN does not match a Vehicle in the database:
  - **Display an error message** "Vehicle with this VIN not found".
- Else (If the Vehicle is not sold and the VIN matches a Vehicle in the database):
  - The Repair Form **display details of the Vehicle**, including VIN, Vehicle Type, Model Year, Model Name, manufacturer, and Color(s).

```
WITH sold_vehicle AS (
SELECT V.VIN, V.vehicle_type, V.model_year, V.model_name,
V.manufacturer
FROM Vehicle AS V
INNER JOIN SalesTransaction AS ST ON V.VIN = ST.VIN)

SELECT V.VIN, V.vehicle_type, V.model_year, V.model_name,
V.manufacturer, C.color
FROM sold_vehicle AS V
INNER JOIN Vehicle_Color AS C
ON V.VIN = C.VIN AND V.VIN = '$VIN';
```

- On the **Repair Form**:
  - **Search Repair** using VIN and Start Date. Check completion date.

```
SELECT VIN, start_date, completion_date
FROM Repair
WHERE VIN = '$VIN' AND start_date = '$start_date';
```

  - If no Repair is found for the Vehicle.
    - While no button is pushed, do nothing.
    - When a button is pushed, ***Start new repair*:**
      - **Lookup the customer** using '$Driver_License_Number' for Individual, or '$Tax_Identification_Number' for Business

```
SELECT customerID
FROM Individual
WHERE Driver_License_Number = '$Driver_License_Number';

SELECT customerID
FROM Business
WHERE Tax_Identification_Number = '$Tax_Identification_Number';
```

      - If the customer is found, stay on repair form
        - Input the odometer_readout, **store the current date** as repair *start date, and description*
        - The corresponding VIN, customerID, userName will be populated.

```
INSERT INTO Repair(VIN, start_date, odometer_readout,
description, userName, customerID)
VALUES('$VIN', CURDATE(), '$odometer_readout',
```

```
'$description', '$userName', '$customerID');
```

- If customer is not found:
  - Show ***Add Customer*** button to add the customer.
  - If ***Add Customer*** is Clicked:
    - Show a dropdown of $customer_type (Front-end harded Individual or Business)

```
SELECT customer_type FROM Customer
```

  - If $customer_type is Individual:
    - Wait for user to input the following: first_name, last_name, driver_license_number, auto populate a customerID

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

    - And input the email, phone_number, street_address, city, state, postal_code

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

  - If $customer_type is Business:
    - Wait for user to input the following: tax_identification_number, business_name, primary_contact_name, primary_contact_title, auto populate customerID

```
INSERT INTO Business (customerID, Tax_Identification_Number, business_name, primary_contact_name, primary_contact_title)
VALUES ('$customerID', '$Tax_Identification_Number', '$business_name', '$primary_contact_name', '$primary_contact_title');
```

    - And input the email, phone_number, street_address, city, state, postal_code

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

  - Click ***Save*** Button to persist the data.
- Enter other repair information:

- Add *Labour Charge*. **Validate** charge is a Float.

```
INSERT INTO Repair (labor_charge)
VALUES ('$labor_charge');
```

- Add *Parts* (VIN, start_date, part_number, vendor_name, price, quantity). **Validate** Quantity is an Int and Price is a Float.

```
INSERT INTO Part(VIN, start_date, part_number,
vendor_name, price, quantity)
VALUES('$VIN', CURDATE(), '$part_number',
'$vendor_name', '$price', '$quantity');
```

- Click ***SAVE*** button. **Save** the new repair form

o If there is an unfinished Repair for the Vehicle (indicated by having qualified repair record but completion_date is null):
While no button is pushed, do nothing.
When a button is pushed, ***update the repair*:**

- Update *Labor Charge*s.
  - If logged in as Service Writer
    - Input new Labor Charge, if new Labor Charge is equal or less than the previous value, throw error message "Labor Charge cannot be less than previous value".
    - Labor Charge must be a Float.

```
UPDATE Repair
SET labor_charge = '$new_labor_charge'
WHERE labor_charge = '$old_labor_charge' AND
'$new_labor_charge' >= '$old_labor_charge';
```

  - If logged in as the Roland Around/Owner
    - Input new Labor Charge, it can be smaller than the the preiovus value.
    - Labor Charge must be a Float.

```
UPDATE Repair
SET labor_charge = '$new_labor_charge'
WHERE labor_charge = '$old_labor_charge';
```

- Add Parts. Wait for user to input *Quantity, Vendor Name, Part Number, Price*:

```
INSERT INTO Part(VIN, start_date, part_number,
vendor_name, price, quantity)
VALUES('$VIN', CURDATE(), '$part_number',
'$vendor_name', '$price', '$quantity');
```

  - Perform validation ：
    - Validate *Quantity* must be integer
    - Validate *Vendor Name* must be STRING
    - Validate *Part Number* must be String
    - Validate *Price* must be Float.
  - If validation fails ：
    - **Display an error message**
  - If all the validations passed:

        - **Populate the fields**
    - Click on ***Complete the repair Button***:
        - The current date is stored on the repair as the completion date

```
INSERT INTO Repair (completion_date)
VALUES (CURDATE());
```

    - Click ***"Save"*** button to save the updates
- If repair is found and finished (indicated by having qualified repair record and completion_date not null):
    - **Display an error message** "Each Vehicle can only have maximum one repair a day".
- If the user clicks ***Cancel***:
    - Jump to Open the Repair Form page

Add/Update manufacturer

Abstract Code

- **View manufacturer**:
  Populate a list of manufacturers

```
SELECT manufacturer FROM manufacturer
```

- While no button is pushed, do nothing.
- When a button is pushed, then do the following:
    - Wait for user to input *manufacturer name*.
    - If ***Add manufacturer***:
      Persist the data.

```
INSERT INTO manufacturer (manufacturer)
VALUES ('$manufacturer')
```

      View manufacturer
      Display manufacturer list with the newly added one.

```
SELECT manufacturer FROM manufacturer
```

    - If ***Update manufacturer***:
      Persist the data.

```
UPDATE manufacturer
SET manufacturer = '$manufacturer'
WHERE manufacturer = '$current_manufacturer'
```

      Update manufacturer
      View manufacturer with the updated one.

```
SELECT manufacturer FROM manufacturer
```

    - If ***Cancel***
      View manufacturer for the current list of manufacturers

```
SELECT manufacturer FROM manufacturer
```