Phase 2 Abstract Code with SQL | CS 6400 - Fall 2019 | **Team 056**

Table of Contents

Log in:

- User clicked ***Log in*** button
- Display input box for user name and password
- While not buttons are pushed, do nothing
- When ***Log in*** button is pushed, do the following

```
SELECT g.password
FROM (
SELECT password, username FROM Salesperson
UNION
SELECT password, username  FROM Owner
UNION
SELECT password, username  FROM Inventory_Clerk
UNION
SELECT password, username  FROM Manager
) g
WHERE g.username = '$UserName';
```

    - If user name and password match with the record, move to the next screen based on user type
    - If user name and password do not match with the record, show message "user name or password incorrect"
- When ***Cancel*** button is pushed
    - Back to **Search Vehicle** screen

Search Vehicle:

- If use is a public user
    - Display the total number of cars available for purchase

```
SELECT SUM(CASE WHEN a.pending_parts='F' THEN 1 ELSE 0 END) AS total_avilable_vehicle
        FROM
        (
                        SELECT v.vin, v.vehicle_description, t.vehicle_type_name, v.model_year,
m.manufacturer_name, v.model_name, v.mileage, v.price_for_sale, c.vehicle_color,
v.sold_date,
                                CASE WHEN (SELECT EXISTS (SELECT * FROM Vehicle v INNER
JOIN PARTS ON v.vin = parts.vin) = 0) THEN 'F' #cars no parts needed
                                WHEN (
                                                (SELECT EXISTS (SELECT * FROM Vehicle v INNER
JOIN PARTS ON v.vin = parts.vin) != 0) #car need parts
                                                AND
                                        (SELECT EXISTS (SELECT * FROM Vehicle v INNER JOIN
PARTS ON v.vin = parts.vin WHERE parts.part_status in ('ordered','received'))=0) # parts status
are not either ordered or received
                                                )
                                                THEN 'F'
                                ELSE 'T'
                        END AS pending_parts
                        FROM Vehicle v
                        INNER JOIN
                        Has_Type t
                        ON v.vin = t.vin
                        INNER JOIN
                        Manufacturered_By m
                        ON v.vin = m.vin
                        INNER JOIN
                        Has_Color c
                        ON v.vin = c.vin
        ) a
        WHERE a.sold_date is NULL;
```

    - Display the below search boxes:
        - Vehicle type (drop down)
        - Manufacturer (drop down)
        - Model year (drop down)
        - Color (drop down)
        - Keyword
    - While not buttons are pushed, do nothing

- When ***Search*** button been pushed

```
#If public user
SELECT b.vin, b.vehicle_type_name, b.model_year, b.manufacturer_name, b.model_name, GROUP_CONCAT(b.vehicle_color
SEPARATOR ',') AS vehicle_colors, b.mileage, b.price_for_sale FROM
(
        SELECT * FROM
        (
                SELECT v.vin, v.vehicle_description, t.vehicle_type_name, v.model_year, m.manufacturer_name,
v.model_name, v.mileage, v.price_for_sale, c.vehicle_color, v.sold_date,
                CASE WHEN (SELECT EXISTS (SELECT * FROM Vehicle v INNER JOIN PARTS ON v.vin =
parts.vin) = 0) THEN 'F' #cars no parts needed
                WHEN (
                        (SELECT EXISTS (SELECT * FROM Vehicle v INNER JOIN PARTS ON
v.vin = parts.vin) != 0) #car need parts
                        AND
                 (SELECT EXISTS (SELECT * FROM Vehicle v INNER JOIN PARTS ON v.vin = parts.vin WHERE
parts.part_status in ('ordered','received'))=0) # parts status are not either ordered or received
                 )
                 THEN 'F'
                ELSE 'T'
                END AS pending_parts
                FROM Vehicle v
                INNER JOIN
                Has_Type t
                ON v.vin = t.vin
                INNER JOIN
                Manufacturered_By m
                ON v.vin = m.vin
                INNER JOIN
                Has_Color c
                ON v.vin = c.vin
        ) a
        WHERE a.vehicle_type_name = '$search_vehicle_type'
        AND a.manufacturer_name = '$search_manufacturer'
        AND a.model_year = '$search_model_year'
        AND a.vehicle_color = '$search_vehicle_color'
        AND a.pending_parts = 'F'
        AND a.sold_date is NULL
        AND (
                a.manufacturer_name LIKE '%$keyword%'
                OR
                a.model_year LIKE '%$keyword%'
                OR
                a.model_name LIKE '%$keyword%'
                OR
                a.vehicle_description LIKE '%$keyword%'
        )
) b
GROUP BY b.vin, b.vehicle_type_name, b.model_year, b.manufacturer_name, b.model_name, b.mileage, b.price_for_sale
ORDER BY b.vin ASC;
```

- If user run **log in** task and logged in as Sales management staff
    - Display the total number of cars available for purchase

```
SELECT SUM(CASE WHEN a.pending_parts='F' THEN 1 ELSE 0 END) AS total_avilable_vehicle
        FROM
        (
                        SELECT v.vin, v.vehicle_description, t.vehicle_type_name, v.model_year,
m.manufacturer_name, v.model_name, v.mileage, v.price_for_sale, c.vehicle_color,
v.sold_date,
                                CASE WHEN (SELECT EXISTS (SELECT * FROM Vehicle v INNER
JOIN PARTS ON v.vin = parts.vin) = 0) THEN 'F' #cars no parts needed
                                WHEN (
                                                (SELECT EXISTS (SELECT * FROM Vehicle v INNER
JOIN PARTS ON v.vin = parts.vin) != 0) #car need parts
                                                AND
                                        (SELECT EXISTS (SELECT * FROM Vehicle v INNER JOIN
PARTS ON v.vin = parts.vin WHERE parts.part_status in ('ordered','received'))=0) # parts status
are not either ordered or received
                                        )
                                        THEN 'F'
                                ELSE 'T'
                        END AS pending_parts
                        FROM Vehicle v
                        INNER JOIN
                        Has_Type t
                        ON v.vin = t.vin
                        INNER JOIN
                        Manufacturered_By m
                        ON v.vin = m.vin
                        INNER JOIN
                        Has_Color c
                        ON v.vin = c.vin
        ) a
        WHERE a.sold_date is NULL;
```

    - Display the below search boxes:
        - Vehicle type (drop down)
        - Manufacturer (drop down)
        - Model year (drop down)
        - Color (drop down)
        - Keyword
        - VIN
    - When no button is not pushed, do nothing

- When ***Search*** button is pushed

```
#If Sales Management Staff
SELECT b.vin, b.vehicle_type_name, b.model_year, b.manufacturer_name, b.model_name, GROUP_CONCAT(b.vehicle_color
SEPARATOR ',') AS vehicle_colors, b.mileage, b.price_for_sale FROM
(
        SELECT * FROM
        (
                SELECT v.vin, v.vehicle_description, t.vehicle_type_name, v.model_year, m.manufacturer_name,
v.model_name, v.mileage, v.price_for_sale, c.vehicle_color, v.sold_date,
                CASE WHEN (SELECT EXISTS (SELECT * FROM Vehicle v INNER JOIN PARTS ON v.vin =
parts.vin) = 0) THEN 'F' #cars no parts needed
                WHEN (
                        (SELECT EXISTS (SELECT * FROM Vehicle v INNER JOIN PARTS ON
v.vin = parts.vin) != 0) #car need parts
                        AND
                 (SELECT EXISTS (SELECT * FROM Vehicle v INNER JOIN PARTS ON v.vin = parts.vin WHERE
parts.part_status in ('ordered','received'))=0) # parts status are not either ordered or received
                 )
                 THEN 'F'
                ELSE 'T'
                END AS pending_parts
                FROM Vehicle v
                INNER JOIN
                Has_Type t
                ON v.vin = t.vin
                INNER JOIN
                Manufacturered_By m
                ON v.vin = m.vin
                INNER JOIN
                Has_Color c
                ON v.vin = c.vin
        ) a
        WHERE a.vehicle_type_name = '$search_vehicle_type'
        AND a.vin = '$search_vin' # add for Sales Management Staff
        AND a.manufacturer_name = '$search_manufacturer'
        AND a.model_year = '$search_model_year'
        AND a.vehicle_color = '$search_vehicle_color'
        AND a.pending_parts = 'F'
        AND a.sold_date is NULL
        AND (
                a.manufacturer_name LIKE '%$keyword%'
                OR
                a.model_year LIKE '%$keyword%'
                OR
                a.model_name LIKE '%$keyword%'
                OR
                a.vehicle_description LIKE '%$keyword%'
        )
) b
GROUP BY b.vin, b.vehicle_type_name, b.model_year, b.manufacturer_name, b.model_name, b.mileage, b.price_for_sale
ORDER BY b.vin ASC;
```

- If user run **log in** task and logged in as Inventory management staff
    - Display the total number of cars available for purchase
    - Display the number of vehicles currently with parts pending

```
SELECT SUM(CASE WHEN a.pending_parts='F' THEN 1 ELSE 0 END) AS total_avilable_vehicle,
            SUM(CASE WHEN a.pending_parts='T' THEN 1 ELSE 0 END) AS
vehicle_with_pending_parts,
        FROM
        (
                        SELECT v.vin, v.vehicle_description, t.vehicle_type_name, v.model_year,
m.manufacturer_name, v.model_name, v.mileage, v.price_for_sale, c.vehicle_color,
v.sold_date,
                                CASE WHEN (SELECT EXISTS (SELECT * FROM Vehicle v INNER
JOIN PARTS ON v.vin = parts.vin) = 0) THEN 'F' #cars no parts needed
                                WHEN (
                                                (SELECT EXISTS (SELECT * FROM Vehicle v INNER
JOIN PARTS ON v.vin = parts.vin) != 0) #car need parts
                                                AND
                                        (SELECT EXISTS (SELECT * FROM Vehicle v INNER JOIN
PARTS ON v.vin = parts.vin WHERE parts.part_status in ('ordered','received'))=0) # parts status
are not either ordered or received
                                        )
                                        THEN 'F'
                                ELSE 'T'
                        END AS pending_parts
                        FROM Vehicle v
                        INNER JOIN
                        Has_Type t
                        ON v.vin = t.vin
                        INNER JOIN
                        Manufacturered_By m
                        ON v.vin = m.vin
                        INNER JOIN
                        Has_Color c
                        ON v.vin = c.vin
        ) a
        WHERE a.sold_date is NULL;
```

    - Display the below search boxes:
        - Vehicle type (drop down)
        - Manufacturer (drop down)
        - Model year (drop down)
        - Color (drop down)
        - Keyword
        - VIN

- When no button is pushed, do nothing
- When ***Search*** button is pushed

```
#If Inventory Management Staff
SELECT b.vin, b.vehicle_type_name, b.model_year, b.manufacturer_name, b.model_name, GROUP_CONCAT(b.vehicle_color
SEPARATOR ',') AS vehicle_colors, b.mileage, b.price_for_sale,
b.pending_parts #added pending_parts for Inventory_Management_Staff
FROM
(
        SELECT * FROM
        (
                SELECT v.vin, v.vehicle_description, t.vehicle_type_name, v.model_year, m.manufacturer_name,
v.model_name, v.mileage, v.price_for_sale, c.vehicle_color, v.sold_date,
                CASE WHEN (SELECT EXISTS (SELECT * FROM Vehicle v INNER JOIN PARTS ON v.vin =
parts.vin) = 0) THEN 'F' #cars no parts needed
                WHEN (
                        (SELECT EXISTS (SELECT * FROM Vehicle v INNER JOIN PARTS ON
v.vin = parts.vin) != 0) #car need parts
                        AND
                 (SELECT EXISTS (SELECT * FROM Vehicle v INNER JOIN PARTS ON v.vin = parts.vin WHERE
parts.part_status in ('ordered','received'))=0) # parts status are not either ordered or received
                 )
                 THEN 'F'
                ELSE 'T'
                END AS pending_parts
                FROM Vehicle v
                INNER JOIN
                Has_Type t
                ON v.vin = t.vin
                INNER JOIN
                Manufacturered_By m
                ON v.vin = m.vin
                INNER JOIN
                Has_Color c
                ON v.vin = c.vin
        ) a
        WHERE a.vehicle_type_name = '$search_vehicle_type'
        AND a.vin = '$search_vin' # add for Sales Management Staff and Inventory_Management_Staff
        AND a.manufacturer_name = '$search_manufacturer'
        AND a.model_year = '$search_model_year'
        AND a.vehicle_color = '$search_vehicle_color'
        AND a.sold_date is NULL
        AND (
                a.manufacturer_name LIKE '%$keyword%'
                OR
                a.model_year LIKE '%$keyword%'
                OR
                a.model_name LIKE '%$keyword%'
                OR
                a.vehicle_description LIKE '%$keyword%'
        )
) b
GROUP BY b.vin, b.vehicle_type_name, b.model_year, b.manufacturer_name, b.model_name, b.mileage, b.price_for_sale,
b.pending_parts
ORDER BY b.vin ASC;
```

- If user run **log in** task and logged in as Manager or Owner:
    - Display the total number of cars available for purchase
    - Display the number of vehicles currently with parts pending

```
SELECT SUM(CASE WHEN a.pending_parts='F' THEN 1 ELSE 0 END) AS total_avilable_vehicle,
            SUM(CASE WHEN a.pending_parts='T' THEN 1 ELSE 0 END) AS
vehicle_with_pending_parts,
        FROM
        (
                        SELECT v.vin, v.vehicle_description, t.vehicle_type_name, v.model_year,
m.manufacturer_name, v.model_name, v.mileage, v.price_for_sale, c.vehicle_color,
v.sold_date,
                                CASE WHEN (SELECT EXISTS (SELECT * FROM Vehicle v INNER
JOIN PARTS ON v.vin = parts.vin) = 0) THEN 'F' #cars no parts needed
                                WHEN (
                                                (SELECT EXISTS (SELECT * FROM Vehicle v INNER
JOIN PARTS ON v.vin = parts.vin) != 0) #car need parts
                                                AND
                                        (SELECT EXISTS (SELECT * FROM Vehicle v INNER JOIN
PARTS ON v.vin = parts.vin WHERE parts.part_status in ('ordered','received'))=0) # parts status
are not either ordered or received
                                        )
                                        THEN 'F'
                                ELSE 'T'
                        END AS pending_parts
                        FROM Vehicle v
                        INNER JOIN
                        Has_Type t
                        ON v.vin = t.vin
                        INNER JOIN
                        Manufacturered_By m
                        ON v.vin = m.vin
                        INNER JOIN
                        Has_Color c
                        ON v.vin = c.vin
        ) a
        WHERE a.sold_date is NULL;
```

    - Display the below search boxes:
        - Vehicle type (drop down)
        - Manufacturer (drop down)
        - Model year (drop down)
        - Color (drop down)
        - Keyword
        - VIN

        - Vehicle Status (drop down) which filters vehicles by Sold Vehicle, Unsold Vehicle and All Vehicles
    - Display a drop down to filter report type (including **Seller History**; **Average Time in Inventory**; **Price Per Condition**; **Parts Statistics**; **Monthly Loan**; **Monthly Sales** and a ***Run Report*** button
        - If user click ***Run Report*** button, run **View Report** task based on the report type selected
    - While not buttons are pushed, do nothing

- When ***Search*** button been pushed

```
#If Managers and Owner
SELECT b.vin, b.vehicle_type_name, b.model_year, b.manufacturer_name, b.model_name, GROUP_CONCAT(b.vehicle_color
SEPARATOR ',') AS vehicle_colors, b.mileage, b.price_for_sale,
b.pending_parts, b.sold
FROM
(
        SELECT * FROM
        (
                SELECT v.vin, v.vehicle_description, t.vehicle_type_name, v.model_year, m.manufacturer_name,
v.model_name, v.mileage, v.price_for_sale, c.vehicle_color, v.sold_date,
                CASE WHEN v.sold_date IS NULL THEN 'F'
        ELSE 'T'
        END AS sold,
                CASE WHEN (SELECT EXISTS (SELECT * FROM Vehicle v INNER JOIN PARTS ON v.vin =
parts.vin) = 0) THEN 'F' #cars no parts needed
                WHEN (
                        (SELECT EXISTS (SELECT * FROM Vehicle v INNER JOIN PARTS ON
v.vin = parts.vin) != 0) #car need parts
                        AND
                 (SELECT EXISTS (SELECT * FROM Vehicle v INNER JOIN PARTS ON v.vin = parts.vin WHERE
parts.part_status in ('ordered','received'))=0) # parts status are not either ordered or received
                 )
                 THEN 'F'
                ELSE 'T'
        END AS pending_parts
        FROM Vehicle v
        INNER JOIN
        Has_Type t
        ON v.vin = t.vin
        INNER JOIN
        Manufacturered_By m
        ON v.vin = m.vin
        INNER JOIN
        Has_Color c
        ON v.vin = c.vin
        ) a
        WHERE a.vehicle_type_name = '$search_vehicle_type'
        AND a.vin = '$search_vin' # add for Sales Management Staff and Inventory_Management_Staff
        AND a.manufacturer_name = '$search_manufacturer'
        AND a.model_year = '$search_model_year'
        AND a.vehicle_color = '$search_vehicle_color'
        AND (
                a.manufacturer_name LIKE '%$keyword%'
                OR
                a.model_year LIKE '%$keyword%'
                OR
                a.model_name LIKE '%$keyword%'
                OR
                a.vehicle_description LIKE '%$keyword%'
        )
) b
GROUP BY b.vin, b.vehicle_type_name, b.model_year, b.manufacturer_name, b.model_name, b.mileage, b.price_for_sale,
b.pending_parts,  b.sold
ORDER BY b.vin ASC;
```

- If user click on an individual result, then run **View Vehicle Detail** task

- When search box is not null, and ***Search*** button been pushed, if there is NO vehicles meet the Criteria:
    - Display message: “Sorry, it looks like we don’t have that in stock!”

Search/Add Customer

- User Clicked ***Search Customer*** button or ***Add Customer*** button from **Add Vehicle Form** or **Sale Order Form**
- When user click ***Search Customer*** button, then:
    - When search box is null, and search button been pushed
        - Display message: “Customer search input cannot be NULL!”
    - When search box is NOT null, and search button been pushed, if there are customer meet the criteria
        - Display the customers with details information

```
#If seach for individual customer:

SELECT c.customer_id, c.street, c.city, c.state, c.postal_code, c.phone_number, c.email_address,
i.drivers_license_number, i.first_name, i.last_name

FROM Customer c

INNER JOIN Individual_person_customer i

ON c.customer_id = i.customer_id

WHERE c.customer_id = ‘$customer_id’;

#If seach for business customer:

SELECT c.customer_id, c.street, c.city, c.state, c.postal_code, c.phone_number, c.email_address,
b.tax_identification_number, b.business_name, b.primary_contact_first_name,
b.primary_contact_last_name

, b.primary_contact_title

FROM Customer c

INNER JOIN Business_customer b

ON c.customer_id = b.customer_id

WHERE c.customer_id = ‘$customer_id’;
```

        - Display ***Select Customer*** button
        - When user click the ***Select Customer*** button, go back to **Add Vehicle Form** or **Sales Order Form**
    - When search box is NOT null, and search button been pushed, if there are NO customer meet the criteria
        - Display message: “No Customer found”
        - Display ***Add Customer*** button

- When user click ***Add Customer*** button, then:
    - Display form to add a new customer
    - Display a drop down to filter individual customer or business customer
    - Input customer information

#If selected individual customer:

INSERT INTO Customer (customer_id, street, city, state, postal_code, phone_number, email_address)

VALUES (‘$customer_id’, ‘$street’, ‘$city’, ‘$state’, ‘$postal_code’, ‘$phone_number’, ‘$email_address’);

INSERT INTO Individual_person_customer (drivers_license_number, first_name, last_name, customer_id)

VALUES ('$drivers_license_number', '$first_name', '$last_name', ‘$customer_id’) ;

#If selected business customer:

INSERT INTO Customer (customer_id, street, city, state, postal_code, phone_number, email_address)

VALUES (‘$customer_id’, ‘$street’, ‘$city’, ‘$state’, ‘$postal_code’, ‘$phone_number’, ‘$email_address’);

INSERT INTO Business_customer (tax_identification_number, business_name, primary_contact_first_name, primary_contact_last_name, primary_contact_title, customer_id)

VALUES (‘$tax_identification_number’, ‘$business_name’, '$primary_contact_first_name', '$primary_contact_last_name', '$primary_conatct_title',’$customer_id’);

    - When user completed customer information, and user clicked ***Save*** button:
    - User input will be checked for against schema/data types and constraints. If an error is found, display "Invalid Input" and highlight the input field causing the issue.
    - Go back **Add Vehicle Form** or **Sales Order Form**

Add vehicle:

- User run **log in** task and logged in as an Inventory Management Staff and clicked ***Add Vehicle*** button
- Display **Add Vehicle Form** which contains search customer box and ***Search Customer*** button for customer search and an ***Add Customer*** button to add customer.
- While no buttons are pushed, do nothing

- When user click ***Search Customer*** or ***Add Customer*** button, run **Search/Add Customer**
- When the seller of the car been selected
    - User lookup and input vehicle type and manufacturer in the database and update if a new manufacturer or vehicle shows up

```
INSERT INTO Manufacturer (manufacturer_name)
VALUES ('$manufacturer_name');
INSERT INTO Vehicle_Type(vehicle_type_name)
VALUES ('$vehicle_type_name');
```

    - User input vehicle details

```
INSERT INTO Vehicle (vin, vehicle_description, model_name, vehicle_condition, mileage,
price_for_sale, buyer_customer_id, seller_customer_id, price_sold, sold_date,
price_purchase, purchase_date)
VALUES (‘$vin’, '$vehicle_description', 'model_name', '$vehicle', '$mileage',
'$price_purchase'*1.25, NULL,'$seller_customer_id', NULL, NULL, '$price_purchase',
'$purchase_date');
```

    - User input will be checked for against schema/data types and constraints. If an error is found, display "Invalid Input" and highlight the input field causing the issue.
- Display ***Complete*** button
- If user clicked ***Complete*** button, update vehicle information
    - Display the **Vehicle Detail** page for the vehicle

Sell Vehicle

- User run **log in** task and logged in as a Sales Management Staff and is on **Vehicle Detail** page and clicked ***Sell Vehicle*** button
- Display **Sales Order Form** which contains search customer box and ***Search Customer*** button for customer search and an ***Add Customer*** button to add customer.
- The **Sales Order Form** will also display the sales price of the vehicle

```
SELECT vin, price_for_sale from Vehicle WHERE vin='$vin';
```

- While no buttons are pushed, do nothing
- When user click ***Search Customer*** or ***Add Customer*** button, run **Search/Add Customer**
- When the Buyer of the car been selected
    - User input *sales date*

```
UPDATE Vehicle

SET sold_date='$sold_date', price_sold =
price_for_sale;
```

- If the customer applied for a loan to purchase the car, user should input loan details to the form

```
#If loan is applied

INSERT INTO Loan (vin, start_month, loan_term, monthly_payment,
interest_rate, downpayment, customer_id)

VALUES (‘$vin’, '$start_month', '$loan_term','$monthly_payment',
'$interest_rate', '$downpayment', ‘$customer_id’);
```

- User input will be checked for against schema/data types and constraints. If an error is found, display "Invalid Input" and highlight the input field causing the issue.

- Display ***Complete*** button
  - User input will be checked for data type consistency as noted in Data Types section. If an error is found, display "Invalid Input Type" and highlight the input field causing the issue.
  - If user clicked ***Complete*** button, update vehicle information
- After the **Sales Order Form** is completed, jump to **Vehicle Search** page

Search/Add Vendor

- User Clicked ***Search Vendor*** button or ***Add Vendor*** button from **Parts Order Form**
- When user click ***Search Vendor*** button, then:
  - When search box is null, and search button been pushed
    - Display message: “Vendor search input cannot be NULL!”
  - When search box is NOT null, and search button been pushed, if there are Vendor meet the criteria
    - Display the customers with detailed information

```
SELECT v.vendor_name, v.phone_number, v.street, v.city, v.state, v.postal_code

FROM Vendor v

WHERE v.vendor_name = '$vendor_name';
```

    - Display ***Select Vendor*** button
    - When user click the ***Select Vendor*** button, go back to **Parts Order Form**
  - When search box is NOT null, and search button been pushed, if there are NO Vendor meet the criteria

        - Display message: “No Vendor found”
        - Display ***Add Vendor*** button
- When user click ***Add Vendor*** button, then:
    - Display form to add a new Vendor
    - User should input Vendor information

```
INSERT INTO Vendor (vendor_name, phone_number, street, city, state,
postal_code)

VALUES ('$vendor', '$phone_number', '$street', '$city', '$state',
'$postal_code');
```

    - When user completed Vendor information, and user clicked ***Save*** button:
    - User input will be checked for against schema/data types and constraints. If an error is found, display "Invalid Input" and highlight the input field causing the issue.
    - Go back to **Parts Order Form**

Order Parts

- User run **log in** task and logged in as an Inventory Management Staff and is on **Vehicle Detail** page and clicked ***Add Part Order*** button
- Display **Parts Order Form** which contains search vendor box and ***Search Vendor*** button for Vendor search and an ***Add Vendor*** button to add Vendor.
- While no buttons are pushed, do nothing
- When user click ***Search Vendor*** or ***Add Vendor*** button, run **Search/Add Vendor**
- When vendor is selected

- User input parts information

```
for each '$part_number', '$part_cost', '$description_of_the_part'

INSERT INTO Parts (vin, burdells_purchase_order_number, part_number,
part_cost, part_status, description_of_the_part)

VALUES ('$vin', '$burdells_purchase_order_number', '$part_number',
'$part_cost', 'ordered', '$description_of_the_part')

UPDATE Vehicle

SET price_for_sale=price_for_sale+'$part_cost'

WHERE vin='$vin'

end for;

INSERT INTO Parts_Order (vin, burdells_purchase_order_number,
vendor_name)

VALUES ('$vin', '$burdells_purchase_order_number', 'vendor_name');
```

- User input will be checked for against schema/data types and constraints. If an error is found, display "Invalid Input" and highlight the input field causing the issue.
- When user complete inputting the part information for one part
  - Display a ***Complete*** button
  - If user push ***Complete*** button, user input will be checked for data type consistency as noted in Data Types section. If an error is found, display "Invalid Input Type" and highlight the input field causing the issue
  - Save the input date and go back to **Vehicle Detail** page, which should be updated with the ordered parts information

Update Parts Status

- User run **log in** task and logged in as an Inventory Management Staff and is on **Vehicle Detail** page and clicked ***Update Status*** button beside one part
- Display dropdown of part status and a ***Confirm*** button:
  - Within the dropdown(ordered/received/installed), only later status can be shown and selected.

```
UPDATE Parts

SET sold_date='$part_status'

WHERE vin='$vin' AND
burdells_purchase_order_number='$burdells_purchase_order_number' AND
part_number='$part_number';
```

- After user selected a status, and click ***Confirm*** button: display the updated information in **Vehicle Detail** form

View Vehicle Detail

- If public user:

```
# If not Inventory Management Staff, not Manager (Public user and Sales Management
Staff)
        SELECT * FROM
        (
                SELECT v.vin, model_year, model_name, mileage, price_sold,
vehicle_description, manufacturer_name, vehicle_type_name,
GROUP_CONCAT(hc.vehicle_color SEPARATOR ',') AS vehicle_colors
                FROM Vehicle v
                LEFT JOIN Manufacturered_By mb ON v.vin = mb.vin
                LEFT JOIN Has_Type ht ON v.vin = ht.vin
                LEFT JOIN Has_Color hc ON v.vin=hc.vin
                GROUP BY v.vin, model_year, model_name, mileage, price_sold,
vehicle_description, manufacturer_name, vehicle_type_name
        ) a
        WHERE a.vin = '$vin';
```

- If user run **log in** task and logged in as Inventory management staff

```
# If Inventory Management Staff
        SELECT a.vin, a.model_year, a.model_name, a.mileage, a.price_sold, a.vehicle_description,
a.manufacturer_name, a.vehicle_type_name, a.vehicle_colors, a.price_purchase, SUM(a.part_cost) as
total_parts_cost
            FROM
                        (
                                SELECT v.vin, model_year, model_name, mileage, price_sold,
vehicle_description, manufacturer_name, vehicle_type_name, GROUP_CONCAT(hc.vehicle_color
SEPARATOR ',') AS vehicle_colors, price_purchase, part_cost
                                FROM Vehicle v
                                LEFT JOIN Manufacturered_By mb ON v.vin = mb.vin
                                LEFT JOIN Has_Type ht ON v.vin = ht.vin
                                LEFT JOIN Has_Color hc ON v.vin=hc.vin
             LEFT JOIN Parts p ON v.vin = p.vin
                                GROUP BY v.vin, model_year, model_name, mileage, price_sold,
vehicle_description, manufacturer_name, vehicle_type_name, price_purchase, part_cost
                        ) a
        WHERE a.vin = '$vin'
        GROUP BY  a.vin, a.model_year, a.model_name, a.mileage, a.price_sold,
a.vehicle_description, a.manufacturer_name, a.vehicle_type_name, a.vehicle_colors,
a.price_purchase;

                #parts detials
SELECT part_number, description_of_the_part, p.burdells_purchase_order_number, part_cost,
part_status, po.vendor_name
FROM Vehicle v
LEFT JOIN Parts p ON v.vin = p.vin
LEFT JOIN Parts_Order po ON p.vin = po.vin
AND p.burdells_purchase_order_number = po.burdells_purchase_order_number
WHERE v.vin = '$vin';
```

- if user clicked ***Update Status*** button: run **Update Parts Status** task
- if user clicked ***Add Part Order*** button: run **Order Parts** task

- If user run **log in** task and logged in as Sales Management staff:

```
# If not Inventory Management Staff, not Manager (Public user and Sales Management Staff)
        SELECT * FROM
        (
                SELECT v.vin, model_year, model_name, mileage, price_sold, vehicle_description, manufacturer_name, vehicle_type_name, GROUP_CONCAT(hc.vehicle_color SEPARATOR ',') AS vehicle_colors
                FROM Vehicle v
                LEFT JOIN Manufacturered_By mb ON v.vin = mb.vin
                LEFT JOIN Has_Type ht ON v.vin = ht.vin
                LEFT JOIN Has_Color hc ON v.vin=hc.vin
                GROUP BY v.vin, model_year, model_name, mileage, price_sold, vehicle_description, manufacturer_name, vehicle_type_name
        ) a
        WHERE a.vin = '$vin';
```

  - Display ***Sell Vehicle*** button
  - If user click ***Sell Vehicle*** button: run **Sell Vehicle** task

- If user run **log in** task and logged in as a Manager or Owner:

```
SELECT a.*,
cust_info_s.name as seller_name,
cust_info_s.address as seller_address,
cust_info_s.phone_number as seller_phone_num,
cust_info_b.name as buyer_name,
cust_info_b.address as buyer_address,
cust_info_b.phone_number as buyer_phone_num
        FROM
                (
                        SELECT v.vin, model_year, model_name, mileage, vehicle_description, manufacturer_name, vehicle_type_name,
GROUP_CONCAT(hc.vehicle_color SEPARATOR ',') AS vehicle_colors, price_purchase, purchase_date, price_sold, sold_date, buyer_customer_id,
seller_customer_id
                        FROM Vehicle v
                        LEFT JOIN Manufacturered_By mb ON v.vin = mb.vin
                        LEFT JOIN Has_Type ht ON v.vin = ht.vin
                        LEFT JOIN Has_Color hc ON v.vin=hc.vin
                        LEFT JOIN Parts p ON v.vin = p.vin
                        GROUP BY v.vin, model_year, model_name, mileage, vehicle_description, manufacturer_name,
vehicle_type_name, price_purchase, purchase_date, price_sold, sold_date, buyer_customer_id, seller_customer_id
                ) a
    INNER JOIN
    (
                SELECT c.customer_id, bc.business_name as name, CONCAT(c.street, c.city, c.state, c.postal_code) as address,
c.phone_number FROM Business_Customer bc JOIN customer c on bc.customer_id = c.customer_id
                UNION
                SELECT c.customer_id, CONCAT(ic.first_name ,' ',ic.last_name)as name, CONCAT(c.street, c.city, c.state, c.postal_code) as
address, c.phone_number FROM Individual_Person_Customer ic JOIN Customer c on ic.customer_id = c.customer_id
                ) cust_info_s
    ON a.seller_customer_id = cust_info_s.customer_id
    LEFT JOIN
    (
                SELECT c.customer_id, bc.business_name as name, CONCAT(c.street, c.city, c.state, c.postal_code) as address,
c.phone_number FROM Business_Customer bc JOIN Customer c on bc.customer_id = c.customer_id
                UNION
                SELECT c.customer_id, CONCAT(ic.first_name ,' ',ic.last_name)as name, CONCAT(c.street, c.city, c.state, c.postal_code) as
address, c.phone_number FROM Individual_Person_Customer ic JOIN Customer c on ic.customer_id = c.customer_id
                ) cust_info_b
     ON a.buyer_customer_id = cust_info_b.customer_id
WHERE a.vin = '$vin';

                #parts details
                SELECT part_number, description_of_the_part, p.burdells_purchase_order_number, part_cost, part_status, po.vendor_name
                FROM Vehicle v
                LEFT JOIN Parts p ON v.vin = p.vin
                LEFT JOIN Parts_Order po ON p.vin = po.vin
                AND p.burdells_purchase_order_number = po.burdells_purchase_order_number
                WHERE v.vin = '$vin';

                #loan details
                SELECT
                l.customer_id,l.start_month,l.loan_term, l.monthly_payment, l.interest_rate, l.downpayment,l.vin
                FROM
                loan l
                JOIN vehicle v
                ON v.vin=l.vin
                WHERE v.buyer_customer_id=l.customer_id
                AND v.vin = '$vin';
```

Reports:

View Seller History

- User run **log in** task and logged in as a manager or owner
- When user select **View Seller History** in the dropdown menu and ***Run Report*** button been pushed

```
SELECT
        IFNULL(bc.business_name,CONCAT(ic.first_name,' ',ic.last_name)) AS Seller_Name,
        COUNT(DISTINCT v.vin) AS Number_Of_Vehicles_Sold,
        AVG(v.price_purchase) AS Average_Purchase_Price,
        AVG(v_parts.no_of_parts) AS Average_Number_Of_Parts_Ordered,
        AVG(v_parts.cost_of_parts) AS Average_Cost_Of_Parts
FROM
        Vehicle v INNER JOIN Customer c ON v.seller_customer_id=c.customer_id
        LEFT JOIN Individual_person_customer ic ON c.customer_id=ic.customer_id
        LEFT JOIN Business_customer bc ON c.customer_id=bc.customer_id
        INNER JOIN
        (SELECT
                v.vin,COUNT(*) AS no_of_parts,SUM(IFNULL(p.part_cost,0)) AS
cost_of_parts
        FROM
                Parts_Order po INNER JOIN Parts p ON p.vin=po.vin AND
p.burdells_purchase_order_number=po.burdells_purchase_order_number
                RIGHT JOIN Vehicle v ON v.vin=po.vin) v_parts
        ON v.vin=v_parts.vin
GROUP BY IFNULL(bc.business_name,CONCAT(ic.first_name,' ',ic.last_name))
ORDER BY Number_Of_Vehicles_Sold DESC;
```

- Highlight seller with a red background if it met with any of the below conditions:
    - Seller who has sold vehicles and shows an average of five or more parts
    - Seller who has sold vehicles and shows the average cost of parts is $500 or more
- Display ***Go Back*** button, go back to **Vehicle Search** page

View Average Time In Inventory

- User run **log in** task and logged in as a manager or owner
- When user select **View Average Time In Inventory** in the dropdown menu and ***Run Report*** button been pushed

```
SELECT
        vt.vehicle_type_name AS Vehicle_Type,
        IFNULL(AVG(DATEDIFF(v.sold_date,v.purchase_date)),'N/A') AS
Average_Time_In_Inventory
FROM
        Has_Type ht INNER JOIN Vehicle v ON ht.vin=v.vin
        RIGHT JOIN Vehicle_Type vt ON vt.vehicle_type_name=ht.vehicle_type_name
WHERE Sold_date IS NOT NULL #is a sold vehicle
GROUP BY vt.vehicle_type_name;
```

- Display ***Go Back*** button, go back to **Vehicle Search** page

View Price Per Condition

- User run **log in** task and logged in as a manager or owner
- When user select **View Price Per Condition** in the dropdown menu and ***Run Report*** button been pushed

```
SELECT t1.Vehicle_Type,Excellent,Very_Good,Good,Fair

FROM

        (SELECT vt.vehicle_type_name AS Vehicle_Type,IFNULL(AVG(v.price_purchase),0) AS
Excellent

        FROM

                Has_Type ht INNER JOIN Vehicle v ON ht.vin=v.vin AND
v.vehicle_condition='Excellent'

                RIGHT JOIN Vehicle_Type vt ON vt.vehicle_type_name=ht.vehicle_type_name

        GROUP BY vt.vehicle_type_name) t1 INNER JOIN

        (SELECT vt.vehicle_type_name AS Vehicle_Type,IFNULL(AVG(v.price_purchase),0) AS
Very_Good

        FROM

                Has_Type ht INNER JOIN Vehicle v ON ht.vin=v.vin AND v.vehicle_condition='Very
Good'

                RIGHT JOIN Vehicle_Type vt ON vt.vehicle_type_name=ht.vehicle_type_name

        GROUP BY vt.vehicle_type_name) t2 ON t1.Vehicle_Type=t2.Vehicle_Type INNER JOIN

        (SELECT vt.vehicle_type_name AS Vehicle_Type,IFNULL(AVG(v.price_purchase),0) AS Good

        FROM

                Has_Type ht INNER JOIN Vehicle v ON ht.vin=v.vin AND v.vehicle_condition='Good'

                RIGHT JOIN Vehicle_Type vt ON vt.vehicle_type_name=ht.vehicle_type_name

        GROUP BY vt.vehicle_type_name) t3 ON t2.Vehicle_Type=t3.Vehicle_Type INNER JOIN

        (SELECT vt.vehicle_type_name AS Vehicle_Type,IFNULL(AVG(v.price_purchase),0) AS Fair

        FROM

                Has_Type ht INNER JOIN Vehicle v ON ht.vin=v.vin AND v.vehicle_condition='Fair'

                RIGHT JOIN Vehicle_Type vt ON vt.vehicle_type_name=ht.vehicle_type_name

        GROUP BY vt.vehicle_type_name) t4 ON t3.Vehicle_Type=t4.Vehicle_Type

ORDER BY t1.Vehicle_Type;
```

- Display ***Go Back*** button, go back to **Vehicle Search** page

View Parts Statistics

- User run **log in** task and logged in as a manager or owner
- When user select **View Parts Statistics** in the dropdown menu and ***Run Report*** button been pushed

```
SELECT
        ven.vendor_name AS Vendor_Name,
        COUNT(*) AS Number_Of_Parts_Supplied,
        SUM(p.part_cost) AS Total_Dollar_Amount_Spent_On_Parts
FROM
        Parts_Order po INNER JOIN Parts p ON
po.burdells_purchase_order_number=p.burdells_purchase_order_number
        INNER JOIN Vendor ven ON po.vendor_name=ven.vendor_name
ORDER BY Total_Dollar_Amount_Spent_On_Parts DESC;
```

- Display ***Go Back*** button, go back to **Vehicle Search** page

View Monthly Loan Income

- User run **log in** task and logged in as a manager or the owner.
- On the **Vehicle Search** screen, when the user selects **Monthly Loan Income** in the dropdown menu and ***Run Report*** button is pushed, run this task.

```
#Use temp tables to generate a list of last 12 months.

CREATE TEMPORARY TABLE IF NOT EXISTS month_list (month varchar(2) NOT NULL, PRIMARY KEY (month));

INSERT INTO month_list

        VALUES ('01'),('02'),('03'),('04'),('05'),('06'),('07'),('08'),('09'),('10'),('11'),('12');

CREATE TEMPORARY TABLE IF NOT EXISTS year_list (year varchar(4)  NOT NULL, PRIMARY KEY (year));

INSERT INTO year_list

        VALUES(EXTRACT(YEAR FROM CURDATE())),(EXTRACT(YEAR FROM CURDATE())-1);

CREATE TEMPORARY TABLE IF NOT EXISTS past_12_months (yearmonth varchar(6) NOT NULL) AS

        (

        SELECT CONCAT(year,month) AS yearmonth

        FROM year_list JOIN month_list

        WHERE

                CONCAT(year,month)>=EXTRACT(YEAR_MONTH FROM DATE_SUB(CURDATE(), INTERVAL 1 YEAR))

                AND CONCAT(year,month)<=EXTRACT(YEAR_MONTH FROM CURDATE())

        ORDER BY CONCAT(year,month) DESC LIMIT 12

        );

#Aggregation for each.

SELECT LEFT(m.yearmonth,4) AS `Year`,RIGHT(m.yearmonth,2) AS 'Month',IFNULL(SUM(monthly_payment),0) AS
Total_Monthly_Payment,IFNULL(SUM(0.01*monthly_payment),0) AS "Mr. Burdell's Share"

FROM

        past_12_months m LEFT JOIN

        (SELECT

                vin,EXTRACT(YEAR_MONTH FROM DATE_ADD(start_month,INTERVAL 1 MONTH)) AS loan_income_start,

                EXTRACT(YEAR_MONTH FROM DATE_ADD(start_month, INTERVAL loan_term MONTH)) AS loan_income_end,

                monthly_payment

        FROM Loan) l

        ON loan_income_start<=yearmonth AND loan_income_end>=yearmonth

GROUP BY LEFT(m.yearmonth,4),RIGHT(m.yearmonth,2)

ORDER BY CONCAT(`Year`,`Month`) DESC;
```

- Display ***Go Back*** button, which goes back to **Vehicle Search** screen.

View Monthly Sales

- User run **log in** task and logged in as a manager or the owner.
- On the **Vehicle Search** screen, when the user selects **Monthly Sales** in the dropdown menu and ***Run Report*** button is pushed, run this task.
- First display the Summary Page of the task

```
SELECT
        EXTRACT(YEAR_MONTH FROM v.sold_date) AS `Year_Month`,
        COUNT(*) AS Vehicles_Sold,
        SUM(v.price_sold) AS Total_Income,
        SUM(v.price_sold-v.price_purchase-v_p.cost_of_parts) AS Total_Net_Income
FROM
        Vehicle v INNER JOIN
        (SELECT v.vin,SUM(p.part_cost) AS cost_of_parts
                FROM Vehicle v INNER JOIN Parts_Order po ON v.vin=po.vin
                        INNER JOIN Parts p ON p.vin=po.vin AND
p.burdells_purchase_order_number=po.burdells_purchase_order_number
                GROUP BY v.vin) v_p
        ON v.vin=v_p.vin
GROUP BY EXTRACT(YEAR_MONTH FROM v.sold_date)
ORDER BY `Year_Month` DESC;
```

- If a year/month has no sales, display nothing for it.
- Display ***Go Back*** button, which goes back to **Vehicle Search** screen.

- When a ***Details*** button for one year/month is clicked, display the Drilldown Report for this year/month.

```
SELECT
        first_name,last_name,
        COUNT(*) AS Vehicles_Sold_This_Month,
        SUM(v.price_sold) AS Total_Sales_This_Month
FROM
        Vehicle v INNER JOIN
        (SELECT first_name,last_name,vin
                FROM
                        Sold_By INNER JOIN Sales_Management_Staff ON
Sold_By.sales_staff_id=Sales_Management_Staff.sales_staff_id
                        INNER JOIN
                        (SELECT first_name,last_name,sales_staff_id FROM Salesperson
                         UNION
                         SELECT first_name,last_name,sales_staff_id FROM Owner) s_detail
                        ON Sales_Management_Staff.sales_staff_id=s_detail.sales_staff_id) s_v
        ON v.vin=s_v.vin
WHERE EXTRACT(YEAR_MONTH FROM v.sold_date)='$Year_Month' AND v.sold_date IS NOT NULL
ORDER BY Vehicles_Sold_This_Month DESC,Total_Sales_This_Month DESC;
```

- The Drilldown Report has a ***Go Back*** button that returns to the Summary Page.