

# Table of Contents

Table of Contents	1
Abstract Code	2
Login	2
Abstract Code	2
Parts Order	2
Abstract Code	2
Vehicle Detail	4
Abstract Code	4
Add Vehicle	8
Abstract Code	8
Add Customer (shared subtask)	10
Abstract Code	10
Sell Vehicle	12
Abstract Code	12
View Sellers History	13
Abstract Code	13
View Average Time in Inventory	14
Abstract Code	14
View Price per Condition	15
Abstract Code	15
View Parts Statistics	16
Abstract Code	16
Monthly Sales Summary	16
Abstract Code	16
Monthly Sales Drilldown	17
Abstract Code	17
Search Vehicles	18
Abstract Code	18
	1

# Abstract Code

## Login

### Abstract Code

- User enters *username* (*\$username*) and *password* (*\$password*) input fields
- When **Log In** button is clicked:
- 

```
SELECT username, user_type FROM app_user WHERE username=
'$username' AND password='$password';
```

- If no User record is found:
  - Go back to **Login** form, with error message:
    - **Error: “No matching user/password”**
- Else
  - Store login information with session variable *\$user* with attributes *\$user.username* and *\$user.user\_type* that also used in rest of application for determining if the user is a InventoryClerk, SalesPerson, Owner, or Manager
  - Go to **Search Screen**

## Parts Order

### Abstract Code

- If *\$user* not defined or *\$user.user\_type* not in ('owner', 'inventory\_clerk')
  - Redirect user to **Search Screen**
- Else:
  - Store VIN of vehicle for which we are adding parts order as *\$vehicle.vin*, display on form as non-editable
  - Display text search field for Vendor Name with **Search** button where name is saved as variable *\$vendor\_name*

```
SELECT name, phone_number, street, city, state, postal_code FROM
vendor WHERE name = '$vendor_name';
```

- If matching record found:
  - Show Name, Phone Number, Street, City, State Postal Code of matching vendor.
  - Provide button to **Associate Vendor** with PartsOrder

- o If click **Associate Vendor**, return to PartsOrder with vendor shown as non-editable
  - o Save vendor name as '\$vendor\_name'
- Else, provide **Add Vendor** button
  - Show editable fields with “Name”, “Street Address”, “City”, “State”, “Postal Code”, “Phone Number” fields and **Save** button. They can click **Save** multiple times until all data validation passes.
  - Vendor is created in database upon clicking **Save**

```
INSERT INTO vendor (name, phone_number, street, city,
state, postal_code) VALUES ('$vendor_name',
'$phone_number', '$street_address', '$city', '$state',
'$postal_code') RETURNING name;
```

- Once pass all data validation, show vendor *Name* shown as non-editable, save as variable '\$vendor\_name'
- o Display **Add Part** button.
  - Upon click of **Add Part** display input fields for “Part Number”, “Price (per unit)”, “Quantity”.
  - The **Add Part** button can be clicked multiple times to add several parts
  - Save locally as array of tuples with values [('\$part\_number', '\$unit\_price', '\$quantity')]
- o Display **Submit** Button
  - Before beginning database transaction:
    - Validate there must be at least 1 part in the Parts Order
    - Validate '\$vendor\_name' is set
    - Validate '\$vehicle.vin' is set
  - In a transaction:
    - Set '\$parts\_order\_number' equal to result of query:

```
INSERT INTO parts_order (vin, ordinal, vendor_name) SELECT
'$vehicle.vin', COUNT(*) + 1, '$vendor_name' from
parts_order WHERE vin='$vehicle.vin' RETURNING
parts_order_number;
```

- For as many tuples of parts as there are, insert them as follows

```
INSERT INTO part (part_number, unit_price, description,
quantity, parts_order_number) VALUES ('$part_number',
'$unit_price', '$description', '$quantity',
'$parts_order_number');
```

- After parts have been added, update related vehicles '\$total\_parts\_price'

```
UPDATE vehicle v
SET
total_parts_price = (
```

```
SELECT COALESCE(SUM(p.quantity * p.unit_price), 0)
FROM part AS p
INNER JOIN
    parts_order AS po
    ON p.parts_order_number = po.parts_order_number
WHERE po.vin = '$vehicle.vin'
)
WHERE v.vin = '$vehicle.vin';
```

- Parts saved in Transaction with rest of Parts Order

## Vehicle Detail

### Abstract Code

- Store '\$vin' of vehicle that was clicked on for the detail view
- Always show public view whether '\$user' is defined or not:
  - Read Vehicle from database, store result as '\$vehicle'

```
SELECT
    v.vin,
    v.vehicle_type,
    v.manufacturer,
    v.model,
    v.description,
    v.model_year,
    v.fuel_type,
    v.horsepower,
    v.purchase_price,
    v.total_parts_price,
    v.customer_seller,
    v.customer_buyer,
    v.inventory_clerk,
    v.salesperson,
    v.sale_date,
    STRING_AGG(vc.color, ', ') AS colors,
    ROUND(
        (1.25 * v.purchase_price) + (1.1 * v.total_parts_price), 2
    ) AS sale_price
FROM vehicle AS v
LEFT JOIN vehicle_color AS vc ON v.vin = vc.vin
WHERE v.vin = '$vin'
GROUP BY
    v.vin,
    v.vehicle_type,
    v.manufacturer,
    v.model,
    v.model_year,
    v.fuel_type,
    v.horsepower,
    v.purchase_price,
```

```
v.total_parts_price,
v.customer_seller,
v.customer_buyer,
v.inventory_clerk,
v.salesperson,
v.sale_date;
```

- Display read-only fields:
  - *VIN* ('\$vehicle.vin')
  - *Vehicle Type* ('\$vehicle.vehicle\_type')
  - *Manufacturer* ('\$vehicle.manufacturer')
  - *Model* ('\$vehicle.model')
  - *Year* ('\$vehicle.model\_year')
  - *Fuel Type* ('\$vehicle.fuel\_type')
  - *Color(s)* ('\$vehicle.colors')
  - *Horsepower* ('\$vehicle.horsepower')
  - *Sale Price* ('\$vehicle.sale\_price')
  - *Description* ('\$vehicle.description') of the vehicle. If defined, otherwise, empty string.
- If '\$user' is defined and '\$user.user\_type' is Owner, Manager, or Inventory Clerk
  - Display **Purchased Price**: '\$vehicle.purchase\_price'
  - Display **Total Cost of Parts**: '\$vehicle.total\_parts\_price'
- If '\$user' is defined and '\$user.user\_type' is Owner, Manager
  - Additionally look up the Customer that sold the vehicle to the dealership '\$vehicle.customer\_seller' and store as '\$seller'

```
SELECT
    cs.phone_number,
    CONCAT(cs.street, ' ', cs.city, ' ', cs.state, ' ', cs.postal_code)
AS address,
    TRIM(COALESCE(CONCAT(b.title, ' ', b.first_name, ' ', b.last_name, ' ',
i.first_name, ' ', i.last_name), '')) AS contact,
    COALESCE(b.business_name, NULL) AS business_name
FROM customer AS cs
LEFT JOIN individual AS i ON cs.tax_id = i.ssn
LEFT JOIN business AS b ON cs.tax_id = b.tin
WHERE cs.tax_id = '$vehicle.customer_seller';
```

- Display **Purchased From**:
  - **Contact**: '\$seller.contact'
  - **Phone Number** '\$seller.phone\_number'
  - **Address** '\$seller.address'
  - **Email** '\$seller.email'
  - If '\$seller.business\_name' is not null:
    - Also display **Business Name**: '\$seller.business\_name'

- Display **Purchased Date**: '\$vehicle.purchase\_date'
- Additionally look up the EmployeeBuyer and store result as '\$inventory\_clerk':

```
SELECT
    CONCAT(
        eb.first_name, ' ', eb.last_name
    ) AS name
FROM app_user as eb
WHERE eb.username = '$vehicle.inventory_clerk';
```

- display as "**Purchased By**: '\$purchaser.name'".
- Additionally display the *Purchase Date* of the vehicle ('\$vehicle.purchase\_date')
- If the vehicle's sale date ('\$vehicle.sale\_date') is not null:
  - Additionally look up the VehicleSeller that sold the vehicle and store as '\$salesperson':

```
SELECT
    CONCAT(
        eb.first_name, ' ', eb.last_name
    ) AS name
FROM app_user as eb
WHERE eb.username = '$vehicle.salesperson';
```

- Display "**Sold By**: '\$salesperson.name'".
- Additionally look up the Customer bought the vehicle and store as '\$customer\_buyer'

```
SELECT
    cs.phone_number,
    CONCAT(cs.street, ' ', cs.city, ' ', cs.state, ' ', cs.postal_code) AS address,
    TRIM(COALESCE(CONCAT(b.title, ' ', b.first_name, ' ', b.last_name, ' ', i.first_name, ' ', i.last_name), '')) AS contact,
    COALESCE(b.business_name, NULL) AS business_name
FROM customer AS cs
LEFT JOIN individual AS i ON cs.tax_id = i.ssn
LEFT JOIN business AS b ON cs.tax_id = b.tin
WHERE cs.tax_id = '$vehicle.customer_buyer';
```

- Display **Sold To**:
  - **Contact**: '\$seller.contact'
  - **Phone Number** '\$seller.phone\_number'
  - **Address** '\$seller.address'
  - **Email** '\$seller.email'
  - If '\$seller.business\_name' is not null:
    - Also display **Business Name**: '\$seller.business\_name'
- Display **Sale Date**: '\$vehicle.sale\_date'
- If '\$user' is defined and '\$user.user\_type' is Owner or InventoryClerk

- Get list of parts

```
SELECT
    p.part_number,
    p.description,
    p.quantity,
    p.unit_price,
    p.status,
    p.parts_order_number,
    po.vendor_name
FROM part AS p
INNER JOIN parts_order AS po ON p.parts_order_number = po.parts_order_number
WHERE po.vin = '$vin'
ORDER BY p.parts_order_number;
```

- Display list of Part(s). For each '\$partorder' we previously mapped to their array of '\$parts' we previously saved, display:
  - PartsOrder: '\$partorder.PartsOrderNumber'
    - For each part in the mapped '\$parts' array, display
      - Part Number: '\$part.part\_number'
      - Description: '\$part.description'
      - Quantity: '\$part.quantity'
      - Unit Price: '\$part.unit\_price'
      - Part Status: '\$part.status'
      - Part Order Id: '\$part.parts\_order\_number'
      - If '\$vehicle.sale\_date' is Null and '\$part.status' is not "installed"
        - Display button **Update Parts Status** on each part which allows user to select from dropdown for the status for the selected Part
          - Display status as drop down
          - Display a "**Save**" button on row of part
          - A Part cannot be changed to a previous status in the ordered list "ordered", "received", "installed". Only display valid options. So a part in ordered will only show "received" or "installed" as options. A part in "received" will only show "installed".
          - Once a part is in the "Installed" state it cannot be updated so don't display drop down.
          - After passing these checks, update part status with:

```
UPDATE part
SET status = 'installed'
WHERE
```

```
parts_order_number =
'$part.parts_order_number' AND part_number =
'$part.part_number';
```

- We should check Part Status again before saving and display an Error message if it has already been updated to a status that makes our update invalid.
  - **Error:** “Cannot Update Status”
- If ‘\$vehicle.sale\_date’ is Null (vehicle is not sold)
  - Display button **Add Parts Order** which will take the user to the **Add Parts Order** form
- If ‘\$user’ is defined and ‘\$user.user\_type’ is Owner or SalesPerson
  - Display button **Sell Vehicle** which will take user to the **Sell Vehicle** form

## Add Vehicle

### Abstract Code

- User clicked on **Add Vehicle** button from **Search Screen** page.
  - If ‘\$user’ is not defined or ‘\$user.user\_type’ is not either Inventory Clerk or Owner
    - Return to **Search Screen** page
  - Else
    - Store ‘\$user.username’ as the ‘\$EmployeeBuyer’
    - Display *search* field to look up for customers.
    - User enters *ssn* or *tin* in to *search* field to look up customers and it is stored as ‘\$tax\_id’
      - When the **Search** button is clicked.
 

```
SELECT tax_id FROM customer WHERE tax_id = '$tax_id';
```
  - If a value is found, it is stored as ‘\$customer’.
  - Else:
    - “Customer not found” message displayed.
    - **Add a New Customer** button is displayed.
    - When the user click on **Add a New Customer** button it displays the **Add Customer** sub-form, which executes the **Add Customer** sub-task.
    - When user successfully add a new customer, continue with the customer’s ‘\$ssn’ or ‘\$tin’ is stored as ‘\$customer’
  - After saving customer identifier in ‘\$customer’
  - Enable input for:



- *Vin* ('\$vin')
- *Vehicle Type* ('\$vehicle\_type')
  - Provide drop down of predefined values
- *Manufacturer* ('\$manufacturer')
  - Provide drop down of predefined values
- *Model* ('\$model')
- *Year* ('\$model\_year')
- *Horsepower* ('\$horsepower')
- *Fuel Type* ('\$fuel\_type')
  - Gas, Diesel, Natural Gas, Hybrid, Plugin Hybrid, Battery, or Fuel Cell
- *Colors* ('\$colors'),
  - Provide drop down multi-select and save as list of values
- *Condition* ('\$condition'):
  - Drop down of values:
    - Excellent, Very Good, Good, Fair
- *Description* ('\$description') (optional, store as NULL if not provided),
- *Purchase Price* ('\$purchase\_price')

Color(s) must be at least one that is mentioned in appendix of project spec

- Manufacturer must be one that is mentioned in the appendix of project spec
- SaleDate is instantiated as NULL and not defined until vehicle is sold
- Before enabling Add Vehicle button, require all required fields to have input
- When user clicks on **Add Vehicle** button
  - Attempt to add new vehicle to the database

```
INSERT INTO vehicle (  
    vin,  
    description,  
    horsepower,  
    model_year,  
    model,  
    manufacturer,  
    vehicle_type,  
    purchase_price,  
    purchase_date,  
    condition,  
    fuel_type,  
    inventory_clerk,  
    customer_seller  
)  
VALUES (  
    '$vin',  
    '$description',
```

```
'$horsepower',  
'$model_year',  
'$model',  
'$manufacturer',  
'$vehicle_type',  
'$purchase_price',  
CURRENT_DATE,  
'$condition',  
'$fuel_type',  
'$inventory_clerk',  
'$customer_seller'  
)  
RETURNING vin;
```

- If insert fails, display error
- Else, Store result as '\$vin' and pass to Vehicle Detail page
- [Vehicle Detail](#) page is displayed

## Add Customer (shared subtask)

### Abstract Code

- User opens the sub-form **Add Customer** from either the **Add Vehicle Form** or the **Sell Vehicle Form** after checking that the customer doesn't exist.
- Instantiate local dictionary '\$customer'
- User selects whether the customer is an individual or a business using a radio button.
- If customer type is 'individual':
  - Set '\$customer\_type' to 'i'
  - SSN ('\$tax\_id'): Ensure the format is 9 digits.
- If customer type is 'business'
  - Set '\$customer\_type' to 'b'
  - TIN ('\$tax\_id')
- For all customers, collect:
  - First Name ('\$first\_name')
  - Last Name ('\$last\_name')
  - Street ('\$street')
  - City ('\$city')
  - State ('\$state')
  - Postal Code ('\$postal\_code')
  - Phone Number ('\$phone\_number'): Ensure it matches a valid phone number pattern like 9292009789. Display text box and only accept 10 digits

- Optional: *Email* (**'\$email'**) (set to NULL if not provided)
- If customer type is 'b', also collect:
  - *Business Name* (**'\$business\_name'**)
  - *Title* (**'\$title'**)
- User clicks the **Submit** button on the **Add Customer** form.
  - The following insertions will be made in a transaction, where we commit all changes or rollback if not all succeed/in the case of an error.

```
INSERT INTO customer (  
    tax_id, customer_type, email, phone_number, street, city, state,  
    postal_code  
) VALUES  
(  
    '$tax_id',  
    '$customer_type',  
    '$email',  
    '$phone_number',  
    '$street',  
    '$city',  
    '$state',  
    '$postal_code'  
) RETURNING *;
```

- If customer\_type = **'i'**

```
INSERT INTO individual (ssn, customer_type, first_name,  
    last_name) VALUES  
('$tax_id', '$customer_type', '$first_name', '$last_name');
```

- Else:

```
INSERT INTO business (  
    tin, customer_type, business_name, title, first_name,  
    last_name  
) VALUES  
('$tax_id', '$customer_type', '$business_name', '$title',  
    '$first_name', '$last_name');
```

- Feedback:
  - If successful:
    - Display a message: **"Customer added successfully."**
    - Return **'\$customer'** variable with value of **'\$tax\_id'**
  - If unsuccessful:
    - If validation error: Display an error message indicating the specific issue (e.g., **Error:** "Missing required fields" or **Error:** "Invalid SSN format").
    - If duplicate record: Display an error message:
      - **Error:** "Customer already exists."

- Keep sub-form open and editable

## Sell Vehicle

### Abstract Code

- User clicks on the **Sell Vehicle** button from the **Vehicle Detail** page.
- - If '\$user' is not defined or '\$user.user\_type' is not either Salesperson or Owner
    - Redirect to the **Search Screen** page
  - Else
    - Store '\$user.username' as the '\$salesperson'
    - Store '\$vehicle' along with all attributes from Vehicle Detail screen including '\$vehicle.vin', '\$vehicle.sale\_price' (which was computed on **Search Screen**) for the vehicle that was selected from Search, '\$vehicle.total\_parts\_price', '\$vehicle.sale\_date', '\$vehicle.customer\_buyer', '\$vehicle.salesperson'
    - User enters ssn or tin in to search field to look up customers and it is stored as '\$tax\_id'
      - When the **Search** button is clicked.

```
SELECT tax_id FROM customer WHERE tax_id = '$tax_id';
```

- If a value is found, it is stored as '\$customer'.
- Else:
  - "Customer not found" message displayed.
  - **Add a New Customer** button is displayed.
  - When the user click on **Add a New Customer** button it displays the **Add Customer** sub-form, which executes the **Add Customer** sub-task.
  - When user successfully add a new customer, continue with the customer's '\$ssn' or '\$tin' is stored as '\$customer'
- After the user clicks on the **Confirm Sale** button
  - Check if '\$vehicle.sale\_date' is null and '\$vehicle.customer\_buyer' attribute is null and '\$vehicle.salesperson' is null
    - If all are still null

```
UPDATE vehicle
SET
    sale_date = CURRENT_DATE,
    customer_buyer = '$customer',
    salesperson = '$salesperson'
WHERE vehicle.vin = '$vin' AND sale_date IS NULL;
```

- If succeeds, Display message “Success”
  - Else, return error from UPDATE query
- Else
  - Display an error message “Error: Sorry, vehicle already sold!”
- Return to **Search Screen**

## View Sellers History

### Abstract Code

- If ‘\$user’ is not defined or ‘\$user.user\_type’ not either Owner or Manager:
  - Redirect to **Search Screen**. The user should not be able to see this screen.
- Else
  - Display Title *Sellers History Report*
  - Display a table with columns: *Name/Business, Total Vehicles Sold, Average Purchase Price, Total Parts Count, Average Parts Price/Vehicle*
  - Rows will be result of query:

```

SELECT
    COALESCE(
        b.business_name, CONCAT(i.first_name, ' ', i.last_name)
    ) AS namebusiness,
    COUNT(DISTINCT v.vin) AS vehiclecount,
    COALESCE(ROUND(AVG(v.purchase_price), 2), 0) AS averagepurchaseprice,
    COALESCE(SUM(p.quantity), 0) AS totalpartscount,
    COALESCE(
        ROUND(
            SUM(p.quantity * p.unit_price) / NULLIF(COUNT(DISTINCT v.vin),
0), 2
        ),
        0
    ) AS averagepartscostpervehiclepurchased,
    CASE
        WHEN
            ROUND(
                SUM(p.quantity * p.unit_price)
                / NULLIF(COUNT(DISTINCT v.vin), 0),
                2
            )
            > 500
            OR SUM(p.quantity) / NULLIF(COUNT(DISTINCT v.vin), 0) > 5
        THEN 'highlight'
        ELSE 'no-highlight'
    END AS highlight
FROM
    vehicle AS v
  
```

```

LEFT JOIN
    parts_order AS po ON v.vin = po.vin
LEFT JOIN
    part AS p ON po.parts_order_number = p.parts_order_number
INNER JOIN
    customer AS cs ON v.customer_seller = cs.tax_id
LEFT JOIN
    individual AS i ON cs.tax_id = i.ssn
LEFT JOIN
    business AS b ON cs.tax_id = b.tin
GROUP BY
    cs.tax_id, b.business_name, i.first_name, i.last_name
ORDER BY
    vehiclecount DESC, averagepurchaseprice ASC;

```

- If tuple final value is 'highlight', highlight the row in red

## View Average Time in Inventory

### Abstract Code

- If \$user is not defined or '\$user.user\_type' is not in (Owner, Manager)
  - Redirect to **Search Screen**
- Else:
  - Display Title: "Average Time in Inventory Report"
  - Display a table with columns: *Vehicle Type*, *Average Days in Inventory*

```

SELECT
    vt.vehicle_type,
    COALESCE(
        AVG(
            DATE_PART(
                'day', v.sale_date::TIMESTAMP -
v.purchase_date::TIMESTAMP
            )
            + 1
        )::VARCHAR,
        'N/A'
    ) AS average_time_in_inventory
FROM (
    SELECT UNNEST(ARRAY[
        'Sedan',
        'Coupe',
        'Convertible',
        'CUV',
        'Truck',
        'Van',
        'Minivan',
        'SUV',
        'Other'
    ]) AS vehicle_type

```

```

) AS vt
LEFT JOIN
    vehicle AS v
    ON vt.vehicle_type = v.vehicle_type AND v.sale_date IS NOT
NULL
GROUP BY vt.vehicle_type
ORDER BY vt.vehicle_type;

```

## View Price per Condition

### Abstract Code

- If '\$user' is not defined or '\$user.user\_type' is not in (Owner, Manager):
  - Redirect back to **Search Screen**
- Else:
  - Display Title: "Purchase Price Per Condition Report"
  - Display table with column headings: *Vehicle Type* and then columns for each condition: '*Excellent*', '*Very Good*', '*Good*', '*Fair*'
  - Rows will be result of query:

```

SELECT
    vt.vehicle_type,
    COALESCE(
        SUM(
            CASE WHEN v.condition = 'Excellent' THEN v.purchase_price
        ELSE 0 END
        ),
        0
    ) AS excellenttotalprice,
    COALESCE(
        SUM(
            CASE WHEN v.condition = 'Very Good' THEN v.purchase_price
        ELSE 0 END
        ),
        0
    ) AS verygoodtotalprice,
    COALESCE(
        SUM(CASE WHEN v.condition = 'Good' THEN v.purchase_price ELSE
        0 END), 0
    ) AS goodtotalprice,
    COALESCE(
        SUM(CASE WHEN v.condition = 'Fair' THEN v.purchase_price ELSE
        0 END), 0
    ) AS fairtotalprice
FROM (
    SELECT UNNEST(ARRAY[
        'Sedan',
        'Coupe',
        'Convertible',
        'CUV',
        'Truck',

```

```
        'Van',  
        'Minivan',  
        'SUV',  
        'Other'  
    ]) AS vehicle_type  
  ) AS vt  
  LEFT JOIN vehicle AS v ON vt.vehicle_type = v.vehicle_type  
  GROUP BY vt.vehicle_type  
  ORDER BY vt.vehicle_type DESC;
```

## View Parts Statistics

### Abstract Code

- If '\$user' is not defined or '\$user.user\_type' is not in (Owner, Manager)
  - Redirect back to **Search Screen**
- Else:
  - Display Title: "Parts Stats Report"
  - Show table with columns *Vendor Name, Parts Count, Total Expense*
  - Rows will be result of query:

```
SELECT  
    vendor.name,  
    SUM(part.quantity) AS totalpartsquantity,  
    SUM(part.quantity * part.unit_price) AS vendortotalexpenditure  
FROM parts_order AS partsorder  
INNER JOIN  
    part AS part  
    ON partsorder.parts_order_number = part.parts_order_number  
INNER JOIN vendor AS vendor ON partsorder.vendor_name = vendor.name  
GROUP BY vendor.name  
ORDER BY vendortotalexpenditure DESC;
```

## Monthly Sales Summary

### Abstract Code

- **Summary** Page
  - Table with columns:
    - *Year, Month, Number Vehicles Sold, Gross Income, Net Income, Drilldown*
    - Rows will be result of following query with a link added as last item that will pass the year and month for that row to the Drilldown page

```
SELECT  
    DATE_PART('year', v.sale_date) AS year_sold,  
    DATE_PART('month', v.sale_date) AS month_sold,  
    COUNT(DISTINCT v.vin) AS numbervehicles,  
    SUM(  

```



```

        ROUND((1.25 * v.purchase_price) + (1.1 * v.total_parts_price),
2)
    ) AS grossincome,
    (
        SUM(ROUND((1.25 * v.purchase_price) + (1.1 *
v.total_parts_price), 2))
        - SUM(v.total_parts_price)
    ) AS netincome
FROM vehicle AS v
WHERE v.sale_date IS NOT NULL
GROUP BY
    DATE_PART('year', v.sale_date),
    DATE_PART('month', v.sale_date)
HAVING
    SUM(ROUND((1.25 * v.purchase_price) + (1.1 * v.total_parts_price),
2)) > 0
ORDER BY year_sold DESC, month_sold DESC;

```

- **Drilldown** Page per Month

- Application will save '\$year\_sold' and '\$month\_sold' data for the row clicked for the drilldown report

## Monthly Sales Drilldown

### Abstract Code

- Application will save '\$year\_sold' and '\$month\_sold' data for the row clicked for the drilldown report
  - Show a table with columns
    - *First Name, Last Name, Vehicles Sold, Total Sales*
    - Rows will be result of following query:

```

SELECT
    au.first_name,
    au.last_name,
    vehiclesold,
    totalsales
FROM
    (
        SELECT
            e.username,
            COUNT(DISTINCT v.vin) AS vehiclesold,
            SUM(
                ROUND(
                    (1.25 * v.purchase_price) + (1.1 *
v.total_parts_price), 2
                )
            ) AS totalsales
        FROM vehicle AS v
        INNER JOIN salesperson AS e ON v.salesperson = e.username
        WHERE

```

```
        EXTRACT(YEAR FROM v.sale_date) = '$year_sold'  
        AND EXTRACT(MONTH FROM v.sale_date) = '$month_sold'  
    GROUP BY e.username  
    ) AS a  
    INNER JOIN app_user AS au ON a.username = au.username  
    GROUP BY au.first_name, au.last_name, vehiclesold, totalsales  
    ORDER BY vehiclesold DESC, totalsales DESC;
```

## Search Vehicles

### Abstract Code

- The following is displayed regardless of whether a logged in User or a public user views the **Search Screen**.

- Display total number of cars available for sale (cars without pending parts)
  - To compute:

```
SELECT COUNT(*)  
FROM vehicle AS v  
LEFT JOIN (  
    SELECT po.vin  
    FROM parts_order AS po  
    INNER JOIN part AS p ON po.parts_order_number =  
    p.parts_order_number  
    WHERE p.status <> 'installed'  
) AS po_not_installed ON v.vin = po_not_installed.vin  
WHERE po_not_installed.vin IS NULL  
AND v.sale_date IS NULL;
```

- Display filter options, drop down with single selection
  - Drop down *Vehicle type*

```
SELECT DISTINCT v.vehicle_type FROM vehicle AS v;
```

- Drop down *Manufacturer*

```
SELECT DISTINCT v.manufacturer FROM vehicle AS v;
```

- Drop down for *Year*

```
SELECT DISTINCT v.model_year FROM vehicle AS v;
```

- Drop Down *Fuel Type*

```
SELECT DISTINCT v.fuel_type FROM vehicle AS v;
```

- Drop down for *Color*

```
SELECT DISTINCT v.color FROM vehicle AS v;
```

- Text field for *Keyword*
- Display a **Search** button
- If '\$user' is not defined
  - Display **Login Page** link
- Else If '\$user' is defined (Privileged User)
  - Display Search by VIN input field.
    - When submitted, submit as all uppercase to match VIN in database
  - If '\$user.user\_type' is "InventoryClerk" or "Owner"
    - Display **Add Vehicle** button
    - When user clicks on **Add Vehicle** button, go to **Add Vehicle** page
  - If '\$user.user\_type' is "Manager" or "Owner" or "Inventory Clerk"
    - Display total number of cars with parts in pending
      - To compute:

```
WITH po_not_installed AS (  
    SELECT po.vin  
    FROM parts_order AS po  
    INNER JOIN part AS p ON po.parts_order_number =  
    p.parts_order_number  
    WHERE p.status <> 'installed'  
)  
SELECT COUNT(*)  
FROM vehicle AS v  
LEFT JOIN po_not_installed ON v.vin = po_not_installed.vin  
WHERE  
    po_not_installed.vin IS NOT NULL  
    AND v.sale_date IS NULL;
```

- If '\$user.user\_type' is 'Manager' or 'Owner'
  - Display VIN input field for search and store input as '\$vin'
  - Set '\$include\_parts\_not\_ready' = TRUE
  - Display three radio buttons with the options of Search in
    - **All**
      - If selected, set '\$filter\_type' = 'unsold'
    - **Sold**
      - If selected, set '\$filter\_type' = 'unsold'
    - **Unsold**
      - If selected, set '\$filter\_type' = 'unsold'
  - Display a list of links for the following reports:
    - **Seller History Report** link.
    - **Average Time in Inventory Report** link.

- **Price Per Condition Report** link.
- **Parts Statistics Report** link.
- **Monthly Sales Report** link.
- If '\$user' is NOT defined (public)
  - Set '\$filter\_type' = 'unsold'
  - Set '\$include\_parts\_not\_ready' = FALSE
  - Set '\$vin' to NULL
- If '\$user' is defined AND '\$user.user\_type' is "sales\_person"
  - Set '\$filter\_type' = 'unsold'
  - Set '\$include\_parts\_not\_ready' = FALSE
  - Display VIN input field for search and store input as '\$vin'
- If '\$user' is defined AND '\$user.user\_type' is in "inventory\_clerk"
  - Set '\$filter\_type' = 'unsold'
  - Set '\$include\_parts\_not\_ready' = TRUE
  - Display VIN input field for search and store input as '\$vin'
- When the **Search** button is clicked

```
SELECT
    vw.vin,
    vw.vehicle_type,
    vw.manufacturer,
    vw.model,
    vw.model_year,
    vw.fuel_type,
    vw.colors,
    vw.horsepower,
    vw.sale_price
FROM (
    SELECT
        v.vin,
        v.vehicle_type,
        v.manufacturer,
        v.model,
        v.description,
        v.model_year,
        v.fuel_type,
        v.horsepower,
        v.purchase_price,
        v.sale_date,
        STRING_AGG(vc.color, ', ') AS colors,
        ROUND(
            (1.25 * v.purchase_price) + (1.1 * v.total_parts_price), 2
        ) AS sale_price
    FROM vehicle AS v
    LEFT JOIN vehicle_color AS vc ON v.vin = vc.vin
    GROUP BY
        v.vin,
        v.vehicle_type,
        v.manufacturer,
        v.model,
        v.model_year,
```

```
        v.fuel_type,  
        v.horsepower,  
        v.purchase_price,  
        v.sale_date  
    ) AS vw  
WHERE  
    (vw.vin NOT IN (  
        SELECT po.vin  
        FROM parts_order AS po  
        INNER JOIN part AS p ON po.parts_order_number = p.parts_order_number  
        WHERE p.status <> 'installed'  
    ) OR '$include_parts_not_read')  
AND (  
    (vw.sale_date IS NULL AND '$filter_type' = 'unsold')  
    OR (vw.sale_date IS NOT NULL AND '$filter_type' = 'sold')  
    OR ('$filter_type' = 'both')  
)  
AND (vw.vehicle_type = '$vehicle_type' OR '$vehicle_type' IS NULL)  
AND (vw.manufacturer = '$manufacturer' OR '$manufacturer' IS NULL)  
AND (vw.model_year = '$model_year' OR '$model_year' IS NULL)  
AND (vw.fuel_type = '$fuel_type' OR '$fuel_type' IS NULL)  
AND (vw.colors LIKE NULL OR NULL IS NULL)  
AND (LOWER(vw.vin) = LOWER('$vin') OR '$vin' IS NULL)  
AND (  
    (vw.manufacturer ILIKE '%$keyword%' OR '%$keyword%' = '%%')  
    OR (vw.model ILIKE '%$keyword%' OR '%$keyword%' = '%%')  
    OR (vw.model_year::TEXT ILIKE '%$keyword%' OR '%$keyword%' = '%%')  
    OR (vw.description ILIKE '%$keyword%' OR '%$keyword%' = '%%')  
)  
ORDER BY vw.vin ASC;
```

- If the search query does not return any vehicles, return an **error** message
  - **“Sorry, it looks like we don’t have that in stock!”**
- Else, the search results will be displayed in a list which every item with the following data:
  - VIN
  - Vehicle Type
  - Manufacturer
  - Model
  - Year
  - Fuel Type
  - Colors (In the same line)
  - Horsepower
  - Sale Price
  - Display **See Details** button
    - When user clicks on **See Details** button it will take the user to **Vehicle Detail** page