# FAll2024 - CS6400

Team 110

## Table of Contents

## Default Search Page

- Show ***User Login***, ***Search Vehicle*** tabs
- Set $user_permission_index with default value public user
- Set an empty list named $List of Vehicles
- Execute **View Vehicle Number** task
- Upon:
  - Click ***User Login*** button - Execute the **User Login** task. Update $user_permission_index, upon:
    - If $user_permission_index is *Inventory Clerk* or *Owner*, pop out ***Add Vehicle*** button
    - If $user_permission_index is *Manager* or *Owner*, pop out ***View reports*** button and ***Filtered search result*** button
    - If $user_permission_index is *Inventory Clerk* or *Manager* or *Owner*, execute **View Vehicle Number** task
    - If $user_permission_index is *Inventory Clerk* or *Manager* or *Salespeople* or *Owner*, pop out ***Search Vehicle by VIN*** button
  - Click ***Search Vehicle*** button - Execute **Search Vehicle by Criteria** task.
  - Click ***Search Vehicle by VIN*** button - Execute **Search Vehicle by VIN** task.
  - Click ***Add Vehicle*** button - Jump to **Add Vehicle** task
  - Click ***Filtered search result*** button - Execute **Filtered Vehicle by Sold Status** task
  - Click ***View reports*** button - Pop-out buttons ***Seller History***, ***Average Time in Inventory***, ***Price Per Condition***, ***Parts Statistics***, ***Monthly Sales***
  - Click ***Seller History*** button - Jump to **Seller History** task
  - Click ***Average Time in Inventory*** button - Jump to **Average Time in Inventory** task
  - Click ***Price Per Condition*** button - Jump to **Price Per Condition** task
  - Click ***Parts Statistics*** button - Jump to **Parts Statistics** task
  - Click ***Monthly Sales*** button - Jump to **Monthly Sales** task
  - Click any item in displayed search results - Jump to **View Select Vehicle** task

## User Login

- Pop out **User Login** form
- User enters input fields with *user name* ($username) and *password* ($password)
- When ***Enter*** button is clicked, find the Logged-in User using $username and $password, and update $user_permission_index

```
$user_permission_index = SELECT
CASE
WHEN Logged_in_User.username IS NOT NULL AND Inventory_Clerk.username IS NOT
NULL AND Salespeople.username IS NOT NULL AND Manager.username IS NOT NULL
THEN 4 -- For Owner
WHEN Logged_in_User.username IS NOT NULL AND Inventory_Clerk.username IS NOT
NULL THEN 1 -- For Clerk
```

```
WHEN Logged_in_User.username IS NOT NULL AND Salespeople.username IS NOT NULL
THEN 2 -- For Salespeople
WHEN Logged_in_User.username IS NOT NULL AND Manager.username IS NOT NULL
THEN 3 -- For Manager
ELSE 0 -- For Other Login User
END
FROM `Logged_in_User`
    LEFT JOIN Inventory_Clerk ON Logged_in_User.username =
    Inventory_Clerk.username
    LEFT JOIN Salespeople ON Logged_in_User.username = Salespeople.username
    LEFT JOIN Manager ON Logged_in_User.username = Manager.username
WHERE Logged_in_User.username = '$username'
    AND Logged_in_User.password = '$password';
```

- If $user_permission_index remains null, ask for user to re-enter *user name* ($username) and *password* ($password),and jump to previous two steps.
- Otherwise, jump back to **Default Search Page** form

## Search Vehicle by Criteria

- User select one or more input fields (include *vehicle type* ('$VehicleType'), *manufacturer*('$VehicleManufacturer'), *year*('$VehicleYear'), *fuel type*('$VehicleFuel'), *color*('$VehicleColor'))
  - For *vehicle type* ('$VehicleType'), *manufacturer*('$VehicleManufacturer'), and *color*('$VehicleColor') using dropdowns

    ```
    SELECT colors FROM `Colors`;
    SELECT manufacturer_name FROM `Manufacturer_Name`;
    SELECT vehicle_type FROM `Vehicle_Type`;
    ```

  - For *year*('$VehicleYear') and *fuel type*('$VehicleFuel'), User will decide what to enter
- User enter input field *keyword* ('$Keyword')
- When ***Search*** button is clicked, scanned, filtered and sorted all Vehicle in the system:

- Generate $List of Vehicles by checking any attributes in Vehicle completely/partially matches $Keyword and all other input attributes match. Sorted $List of Vehicles using Vehicle attributes Vehicle.VIN in ascending order.

```
$List_Vehicle=SELECT
    VIN, vehicle_type, manufacturer_name, model_name,
    model_year, fuel_type, horsepower, sale_price,
    GROUP_CONCAT(Vehicle_color.color ORDER BY Vehicle_color.color SEPARATOR
    ',')
FROM `Vehicle` INNER JOIN Vehicle_color on Vehicle.VIN = Vehicle_color.VIN
WHERE (Vehicle.vehicle_type='$VehicleType' or '$VehicleType' IS NULL)
AND (Vehicle.manufacture_name='$VehicleManufacturer' or
'$VehicleManufacturer' IS NULL)
AND (Vehicle.model_year='$VehicleYear' or '$VehicleYear' IS NULL)
AND (Vehicle.fuel_type='$VehicleFuel' or '$VehicleFuel' IS NULL)
AND (('$Keyword' IS NULL)
    OR Vehicle.vehicle_type LIKE CONCAT('%', '$Keyword', '%'))
    OR Vehicle.manufacture_name LIKE CONCAT('%', '$Keyword', '%'))
    OR Vehicle.model_year LIKE CONCAT('%', '$Keyword', '%'))
    OR Vehicle.description LIKE CONCAT('%', '$Keyword', '%'))
```

```
HAVING SUM(CASE WHEN Vehicle_color.color='$VehicleColor' Then 1 Else 0 END) >
0
ORDER BY VIN ASC;
```

- If $user_permission_index represents *public_user* or *salespeople*, then filter sold vehicles by buy relationship as well as pending vehicles by Part.status, and display the result:

```
SELECT * FROM { --exclude vehicles that have parts not ready
SELECT $List_Vehicle.*
FROM $List_Vehicle
LEFT JOIN Part ON $List_Vehicle.VIN = Part.VIN
GROUP BY $List_Vehicle.VIN
HAVING SUM(CASE WHEN Part.status = 0 THEN 1 ELSE 0 END) = 0;
}
LEFT JOIN Buy ON $List_Vehicle.VIN = Buy.VIN
WHERE Buy.VIN IS NULL; --exclude vehicles that are sold
```

- If $user_permission_index represents *Inventory_clerk*, then filter sold vehicles by buy relationship and display the result:

```
SELECT * FROM $List_Vehicle
LEFT JOIN Buy ON $List_Vehicle.VIN = Buy.VIN
WHERE Buy.VIN IS NULL; --exclude vehicles that are sold
```

- If $user_permission_index represents *manager* or *owner*, then display $List_Vehicle
- If no result is displayed, display error message: "Sorry, it looks like we don't have that in stock!"
- Select an individual result - Jump to **View Select Vehicle** task.

## Search Vehicle by VIN

- If $user_permission_index represents neither *inventory_clerk* nor *salespeople* nor *manager* nor *owner*, then exit this task
- User enter input field *vin* ('$VIN')
- Check whether all character in $VIN are either number or upper case english character
- When ***Search*** button is clicked, find the matching vehicle:

```
$List_Vehicle=SELECT
    Vehicle.VIN, Vehicle.vehicletype, Vehicle.manufacturer,
    Vehicle.model_name, Vehicle.model_year, Vehicle.fuel_type,
    Vehicle.horsepower, Vehicle.sale_price
    GROUP_CONCAT(Vehicle_color.color ORDER BY Vehicle_color.color SEPARATOR
    ',')
FROM `Vehicle` LEFT JOIN Vehicle_color ON Vehicle.VIN = Vehicle_color.VIN
WHERE (Vehicle.VIN=$VIN)
GROUP BY Vehicle.VIN;
```

- If $user_permission_index represents *salespeople*, then filter sold vehicles by buy relationship as well as pending vehicles by Part.status, and display the result:

```
SELECT * FROM { --exclude vehicles that have parts not ready
SELECT $List_Vehicle.*
FROM $List_Vehicle
LEFT JOIN Part ON $List_Vehicle.VIN = Part.VIN
GROUP BY $List_Vehicle.VIN
```

```
HAVING SUM(CASE WHEN Part.status = 0 THEN 1 ELSE 0 END) = 0;
}
LEFT JOIN Buy ON $List_Vehicle.VIN = Buy.VIN
WHERE Buy.VIN IS NULL; --exclude vehicles that are sold
```

- If $user_permission_index represents *Inventory_clerk*, then filter sold vehicles by buy relationship and display the result:

```
SELECT * FROM $List_Vehicle
LEFT JOIN Buy ON $List_Vehicle.VIN = Buy.VIN
WHERE Buy.VIN IS NULL; --exclude vehicles that are sold
```

- If $user_permission_index represents *manager* or *owner*, then display $List_Vehicle
- If no result is displayed, display error message: "Sorry, it looks like we don't have that in stock!"
- Select an individual result - Jump to **View Select Vehicle** task.

## View Vehicle Number

- Compute and display $Sale_Avail, the number of vehicle that are ready to be sold, which is calculated by filtering Vehicle with pending parts by Part.status

```
$Sale_Avail = SELECT COUNT(*) FROM {
SELECT Vehicle.*
FROM Vehicle
LEFT JOIN Part ON Vehicle.VIN = Part.VIN
GROUP BY Vehicle.VIN
HAVING SUM(CASE WHEN Part.status = 0 THEN 1 ELSE 0 END) = 0;
}
LEFT JOIN Buy ON Vehicle.VIN = Buy.VIN
WHERE Buy.VIN IS NULL; --exclude vehicles that are sold
```

- Check $user_permission_index, if represents *Inventory_clerks*, *Managers*, and *Owner*, then display $Pend_Num=$Unsold_Num-$Sale_Avail
  - $Unsold_Num, the number of vehicle that are not yet sold, which is calculated by filtering Vehicle with buy relationship

```
$Unsold_Num = SELECT COUNT(*) FROM Vehicle
LEFT JOIN Buy ON Vehicle.VIN = Buy.VIN
WHERE Buy.VIN IS NULL;
```

## Filter Vehicle by Sold Status

- This task will only be available if $user_permission_index represents *manager* or *owner*.
- Read $List_Vehicle
- Pop-out three buttons, ***Filtered by sold vehicles***, ***Filtered by unsold vehicles***, ***No filtering***
- If ***Filtered by sold vehicles*** button is clicked, display $Sold_List, the found vehicles that have been sold. Filter Vehicle that has buy relationship.

```
$Sold_List = SELECT $List_Vehicle.* FROM $List_Vehicle
INNER JOIN Buy ON $List_Vehicle.VIN = Buy.VIN;
```

- If ***Filtered by unsold vehicles*** button is clicked, display $Unsold_List, the found vehicles that haven't been sold. Filter Vehicle that has no buy relationship.

```
$Unsold_List = SELECT $List_Vehicle.* FROM $List_Vehicle
LEFT JOIN Buy ON $List_Vehicle.VIN = Buy.VIN
WHERE Buy.VIN IS NULL;
```

- If ***No filtering*** button is clicked, clear display $List_Vehicle

# View Select Vehicle

- Read $user_permission_index and $VIN which should be available when user click the vehicle in results of search
- Pop up **Vehicle Detail Form** and display Vehicle.VIN, Vehicle.type, Vehicle.manufacturer, Vehicle.model, Vehicle.year, Vehicle.fuel type, Vehicle.color, Vehicle.horsepower, Vehicle.sale price, Vehicle.description

```
SELECT Vehicle.VIN, Vehicle.vehicle_type, Vehicle.manufacturer_name,
       Vehicle.model_name, Vehicle.model_year, Vehicle.fuel_type,
       Vehicle.horsepower, Vehicle.sale_price, Vehicle.description
       GROUP_CONCAT(Vehicle_color.color ORDER BY Vehicle_color.color
       SEPARATOR ',')
FROM Vehicle
JOIN Vehicle_color ON Vehicle.VIN= Vehicle_color.VIN
WHERE Vehicle.VIN=$VIN
GROUP BY Vehicle.VIN;
```

- If $user_permission_index represents *salespeople* or *Owner*:
  - Pop-out ***Sell the car*** button
  - If ***Sell the car*** button is clicked, jump to **Sell Vehicle** task
- If $user_permission_index represents *inventory clerks* or *Manager* or *Owner*, execute **View Purchase Price and Parts** subtask
- If $user_permission_index represents *Manager* or *Owner*, execute **View Vehicle Sell and Buy History** subtask

## View Purchase Price and Parts subtask

- Since this task is performed in **Vehicle Detail Form**, the VIN of the vehicle should be available
- Extract the $Purchase_price from Vehicle.purchase price and display it:

```
$Purchase_price =  SELECT purchase_price FROM Vehicle
WHERE Vehicle.VIN=$VIN
```

- Extract the $Total_part_cost by summing all Parts Order.total cost for this Vehicle.VIN, and display it:

```
$Total_part_cost = SELECT SUM(Parts_Order.total_cost)
FROM Parts_Order
WHERE Parts_Order.VIN = '$VIN';
```

- Collect and display all parts order that related to this vehicle
- Display Part.vendor part number, Part.description, Part.unit price, Part.quantity, Part.status, Parts Order.order number, Vendor.name, which is filtered by the Vehicle.VIN ($VIN)

```
SELECT Part.vendor_part_number, Part.description, Part.unit_price,
       Part.quantity, Part.status, Part.oder_number, Parts_Order.vendor_name
FROM Part
JOIN Parts_Order ON Part.order_number = Parts_Order.order_number
WHERE Part.VIN = '$VIN';
```

- If $user_permission_index is inventory clerks or Owner:
  - If the status of any parts is clicked, execute **Update Parts Status** task.
  - Pop-out ***Add parts order*** button.
  - If ***Add parts order*** button is clicked, jump to **Add Parts Order** task

## View Vehicle Sell and Buy History subtask

- Since this task is performed in **Vehicle Detail Form**, the $VIN of the vehicle should be available
- Find the seller from the sell relationship which links Vehicle, Inventory Clerk, and Customer
- Display the seller's contact information from Customer, and identify if the customer is Individual or Business, and also display the individual/business information
  - Display Customer.address,Customer.phone number, Customer.email address
  - Display Individual.name,Business.primary contact, Business.business name
  - Display Vehicle.VIN

```
SELECT Vehicle.VIN,
       Individual.first_name, Individual.last_name,
       Business.primary_contact_first_name,
       Business.primary_contact_last_name, Business.primary_contact_title,
       Business.business_name, Customer.address, Customer.phone_number,
       Customer.email_address
FROM Vehicle
LEFT JOIN Individual ON Vehicle.seller_customer_ID = Individual.customer_ID
LEFT JOIN Business ON Vehicle.seller_customer_ID = Business.customer_ID
LEFT JOIN Customer ON Vehicle.seller_customer_ID = Customer.customer_ID
WHERE Vehicle.VIN = $VIN;
```

- Find and display the inventory clerk's name from the sell relationship which links Vehicle, Inventory Clerk, and Customer
- Find and display the first name and last name of inventory clerk from the Logged-in User
  - Display Logged-in User.name

```
SELECT Logged_in_User.first_name, Logged_in_User.last_name
FROM Vehicle
LEFT JOIN Logged_in_User ON Vehicle.inventoryclerk_username =
Logged_in_User.username
WHERE Vehicle.VIN = $VIN
```

- If this vehicle is sold, find and display the information of buyer and salespeople
  - Find the buyer from the buy relationship which links Vehicle, salespeople, and Customer
  - Display the buyer's contact information from Customer, and identify if the customer is Individual or Business, and also display the individual/business information
    - Display Customer.address,Customer.phone number, Customer.email address

* Display Individual.name,Business.primary contact, Business.business name
* Display Logged-in User.name, buy.sale date

```
SELECT Buy.sale_date,
       Individual.first_name, Individual.last_name,
       Business.primary_contact_first_name,
       Business.primary_contact_last_name, Business.primary_contact_title,
       Business.business_name, Customer.address, Customer.phone_number,
       Customer.email_address, Logged_in_User.first_name,
       Logged_in_User.last_name
FROM Buy
LEFT JOIN Individual ON Buy.buyer_customer_ID = Individual.customer_ID
LEFT JOIN Business ON Buy.buyer_customer_ID = Business.customer_ID
LEFT JOIN Customer ON Buy.buyer_customer_ID = Customer.customer_ID
LEFT JOIN Logged_in_User ON Buy.salespeople_username = Logged_in_User.username
WHERE Buy.VIN = $VIN;
```

## Update Parts Status

- From the **Vehicle Detail Form**, information of the vehicle and related parts should be displayed for *inventory clerk*, *manager*, or *owner*
- For each part, an ***Update*** button will be available.
- When the user clicks the button, given the $order_number (Parts Order.order number), and $vendor_part_number (Part.vendor part number) of the part, update the status of the part (Part.status)
  - If the part is ordered, update it to received, if it's received, update it to installed

```
UPDATE Part
SET Part.status = CASE
    WHEN Part.status = 'ordered' AND @new_status = 'received' THEN 'received'
    WHEN Part.status = 'received' AND @new_status = 'installed' THEN
    'installed'
    ELSE Part.status -- No change if the transition is invalid
END
WHERE Part.order_number = $order_number
AND Part.vendor_part_number = $vendor_part_number;
```

- Since the sale price will be meaningful only when the parts are all ready, if in previous step the status is updated to installed, update Vehicle.sale price by summing 125% of Vehicle.purchase price and 110% of all part cost (sum of the Parts Order.total cost) related to this vehicle ($VIN)

```
UPDATE Vehicle
SET Vehicle.sale_price = 1.25 * Vehicle.purchase_price + 1.10 * (
    SELECT IFNULL(SUM(Parts_Order.total_cost), 0)
    FROM Parts_Order
    WHERE Parts_Order.VIN = $VIN
)
WHERE Vehicle.VIN = $VIN;
```

## Add Vehicle

- After the user logged-in, if $user_permission_index represents *Inventory Clerk* or *Owner*, then the user will be provided with ***Add Vehicle*** button

- If ***Add Vehicle*** button is clicked, then the user will be directed to **Add Vehicle Form**
- Execute **Search Customer** task to acquire $Customer_ID
- If $Customer_ID is none, execute **Add Customer** task
- Pop-out input fields *vin* ($VehicleVIN), *vehicle type* ($VehicleType), *manufacturer name* ($VehicleManu), *model name* ($VehicleModel), *model year* ($VehicleYear), *fuel type*('$VehicleFuel') *color* ($VehicleColor), *horsepower* ($VehiclePow), *description* ($VehicleDes), *purchase price* ($VehiclePurchasePrice), *vehicle condition* ($VehicleCondition),
- User filled in all required input fields:
  - For *vehicle type* ($VehicleType), *manufacturer name* ($VehicleManu) using dropdowns, with options retrieved from Manufacturer_Name and Vehicle_Type

```
SELECT manufacturer_name FROM `Manufacturer_Name`;
SELECT vehicle_type FROM `Vehicle_Type`;
```

  - For *color* ($VehicleColor) can be selected for mutiple times using dropdowns with options retrieved from Colors

```
SELECT colors FROM `Colors`;
```

  - For *vin* ($VehicleVIN), *model name* ($VehicleModel), *model year* ($VehicleYear), *fuel type*('$VehicleFuel'), *horsepower* ($VehiclePow), *description* ($VehicleDes), *purchase price* ($VehiclePurchasePrice), and *vehicle condition* ($VehicleCondition), user will manually fill-in these fields
  - Automatically generate for attributes Vehicle.sale price ($VehicleSalePrice=125% $VehiclePurchasePrice), Vehicle.purchase date ($VehiclePurchaseDate=default of the add vehicle date)
- Add the vehicle in Vehicle using the vehicle information, together with the seller ID $CustomerID and the username $ClerkID of the user who adds this vehicle

```
INSERT INTO Vehicle (VIN, model_name, model_year, fuel_type, horsepower,
description, sale_price, purchase_price, vehicle_condision, purchase_date,
seller_customer_ID, inventoryclerk_username, vehicle_type, manufacturer_name)
VALUES ('$VehicleVIN', '$VehicleModel', '$VehicleYear', '$VehicleFuel',
'$VehiclePow', '$VehicleDes', '$VehicleSalePrice', '$VehiclePurchasePrice',
'$VehicleCondition', '$VehiclePurchaseDate', '$CustomerID', '$ClerkID',
'$VehicleType', '$VehicleManu');
```

- For each color $EachColor in $VehicleColor, add it to Vehicle.color:

```
INSERT INTO Vehicle_color (VIN, color)
VALUES ('$VehicleVIN', '$EachColor');
```

- Jump to **View Select Vehicle** task with the input $VIN (Vehicle.VIN)

## Search Customer

- This task will be available when *inventory clerk* or *owner* try to search custormer on **Add Vehicle Form** as well as when *salespeople* or *owner* try to search custormer on **Sell Vehicle Form**
- Find the customer when user input either by SSN ('$SSN') or taxID ('$ITIN')
- If the customer is individual,
  - User enter input field *ssn* ('$SSN')

  - Find and display if any Individual.SSN match $SSN, then update $Customer_ID and exit this task

```
$Customer_ID = SELECT customer_ID FROM `Customer`
INNER JOIN Individual WHERE Individual.SSN = '$SSN';
```

- If the customer is business,
  - User enter input field *itin* ('$ITIN')
  - Find and display if any Business.ITIN match $ITIN, then update $Customer_ID and exit this task

```
$Customer_ID = SELECT customer_ID FROM `Customer`
INNER JOIN Business WHERE Business.ITIN = '$ITIN';
```

## Add Customer

- This task will be available when *inventory clerk* or *owner* try to add custormer on **Add Vehicle Form** as well as when *salespeople* or *owner* try to add custormer on **Sell Vehicle Form**
- Pop-out input fields *address* ($CustomerAddress), *phone number*($CustomerPhone), *email address*($CustomerEmail), *individual or business* ($CustomerType)
- User enter input fields *address*, *phone number*, *email address*, *individual or business*
- User enter input fields depending on individual/business:
  - If the customer is an individual, pop-out input fields *first name*($IndFirstName), *last name*($IndLastName), and *ssn*($IndSSN)
  - Else, pop-out *itin*($BusITIN), *business name* ($BusName), *primary contact first name*($BusFirstName), *primary contact last name*($BusLastName),and *primary contact title* ($BusTitle)
- Add the customer to Customer

```
INSERT INTO Customer (address, phone_number, email_address)
VALUES ('$CustomerAddress', '$CustomerPhone', '$CustomerEmail');
```

- Determine the $Customer_ID based on the number of items of Customer

```
$Customer_ID = SELECT COUNT(*) FROM Customer;
```

- Add the customer to Individual or Business
  - If Individual, then

```
INSERT INTO Individual (SSN, first_name, last_name, customer_ID)
VALUES ('$IndSSN', '$IndFirstName', '$IndLastName', '$Customer_ID');
```

  - If Business, then

```
INSERT INTO Business (ITIN, primary_contact_first_name,
primary_contact_last_name, primary_contact_title, business_name,
customer_ID)
VALUES ('$BusITIN', '$BusFirstName', '$BusLastName', '$BusTitle',
'$BusName', '$Customer_ID');
```

- Update $Customer_ID and send it back to the form that leads to this function, can be either **Add Vehicle Form** or **Sell Vehicle Form**

## Sell Vehicle

- This task will be performed when *salespeople* or *owner* click the ***Sell the car*** button on **Vehicle Detail Form**, so that $VehicleVIN should be available
- The username of current user $UserName should also be available
- Pop out **Sell Vehicle Form**, providing input fields *ssn* ($SSN) and *itin* ($ITIN), and ***Search Customer*** and ***Confirm the sale*** buttons
- Execute **Search Customer** task using input fields to update $Customer_ID. If $Customer_ID is none, execute **Add Customer** task
- If ***Confirm the sale*** button is clicked, collect the date of today as $SaleDate (buy.sale date) as well as the buyer as $Customer_ID, and record the sale transaction in buy.

```
INSERT INTO Buy (VIN, buyer_customer_ID, salespeople_username, sale_date)
VALUES ('$VehicleVIN', '$customer_ID', '$UserName', '$SaleDate');
```

## Add Parts Order

- This task will be performed when *inventory clerks* or *owner* click the ***Add parts order*** button on **Vehicle Detail Form**, so $VehicleVIN should be available
- Determine the $PartsOrderNum of this order by checking the number of orders in Parts Order related to this Vehicle.VIN

```
$PartsOrderNum = SELECT COUNT(*) FROM Parts_Order WHERE Parts_Order.VIN = $VIN
```

- Execute **Search Vendor** task to update $VendorID. If $VendorID is none, execute **Add Vendor** task
- Add items into Parts_Order using Vehicle.VIN, $PartsOrderNum+1, $VendorID, with default of totalcost as 0

```
INSERT INTO Parts_Order (VIN, order_number, total_cost, vendor_name)
VALUES ('$VehicleVIN', '$PartsOrderNum'+1, '0', '$VendrID');
```

- Pop-out ***Add part*** and ***Finish Parts Order*** buttons. Until ***Finish Parts Order*** button is clicked, execute the following loop:
  - Pop-out (or empty) input fields for *status* ($PartStatus), *description* ($PartDes), *vendor's part number* ($PartNum), *unit price* ($PartPrice), *quantity* ($PartQuantity) for user to fill in
  - When ***Add part*** button is clicked, add the part in Parts_Order,

```
INSERT INTO Part (VIN, order_number, vendor_part_number, status,
description, unit_price, quantity)
VALUES ('$VehicleVIN', '$PartsOrderNum', '$PartNum', '$PartStatus',
'$PartDes', '$PartPrice', '$PartQuantity');
```

- When ***Finish Parts Order*** button is clicked, compute the total cost of this part order and update it in Parts_Order.total cost

```
UPDATE Parts_Order
SET Parts_Order.total_cost = (
    SELECT SUM(Part.unit_price * Part.quantity)
    FROM Part
    WHERE Part.order_number = Parts_Order.order_number
);
```

## Search Vendor

- This task will be available when *inventory clerk* or *owner* try to search vendor on **Add Parts Order Form**
- User enter input field *name* ('$name')
- Find vendor using Vendor.name, update $Vendor_ID and exit

```
$Vendor_ID = SELECT name FROM `Vendor` WHERE Vendor.name = '$name';
```

## Add Vendor

- This task will be available when *inventory clerk* or *owner* try to add vendor on **Add Parts Order Form**
- Pop-out input fields *address*('$VendorAddress'), *phone number*('$VendorPhone'), *vendor name*('$VendorName')
- User enter input fields *address*, *phone number*, *vendor name*
- Add the vendor in Vendor using the above information and update $Vendor ID=$VendorName

```
INSERT INTO Vendor (name, address, phone_num)
VALUES ('$VendorName', '$VendorAddress', '$VendorPhone');
```

## Seller History

- This task will be performed when *manager* or *owner* clicked on ***Seller History*** button from **Default Search Page**
- Find all seller in Customer that has sold vehicle to this system which has the sell relationship to the Vehicle
  - Find the related Part for each Vehicle by Vehicle.VIN
  - Group the results for each seller:
    - Count the total number of vehicles as $num_vehicle
    - Calculate average Vehicle.purchase price as $avg_purchase
    - Sum all Part.quantity as $total_part_quantity
    - Calculate average of (Part.quantity*Part.unit price) as $total_part_cost
  - Sort the results ordered by $num_vehicle and $avg_purchase descending
  - Display $Individual_name, $Business_name, $num_vehicle, $avg_purchase, $total_part_quantity, and $total_part_cost on **Seller History** form.

```
SELECT
    CONCACT(Individual.first_name, Individual.last_name) AS
    '$Individual_name',
    Business.business_name AS '$Business_name',
    COUNT(Vehicle.VIN) AS '$num_vehicle',
    AVG(Vehicle.purchase_price) AS '$avg_purchase$',
    SUM(Part.quantity) AS '$total_part_quantity',
    AVG(Part.quantity*Part.unit_price) AS '$total_part_cost'
FROM
    Customer
JOIN
    Vehicle ON Customer.customer_ID = Vehicle.seller_customer_ID
JOIN
```

```
    Part ON Part.VIN = Vehicle.VIN
LEFT JOIN
    Individual ON Individual.customer_ID = Customer.customer_ID
LEFT JOIN
    Business ON Business.customer_ID = Customer.customer_ID
GROUP BY
    Customer.customer_ID
ORDER BY COUNT(Vehicle.VIN) DESC,
         AVG(Vehicle.purchase_price) ASC;
```

- For each seller, if $total_part_quantity/$num_vehicle for this item is equal or larger than 5, or if $total_part_cost/$num_vehicle is equal or larger than $500, then show this seller with red background.

## Average Time in Inventory

- For each Vehicle Type, find the time average of its inventory time (buy.sale date-Vehicle.purchase date)
  - Find all Vehicle related to certain Vehicle Type.vehicle type
  - For the result, find if that Vehicle has buy relationship
  - Calculated for average (buy.sale date-Vehicle.purchase date) for that Vehicle Type.vehicle type
  - Display Vehicle Type.vehicle type and average inventory time on **Average Time in Inventory** form.
  - If no average inventory time, display Vehicle Type.vehicle type and N/A

```
SELECT
    Vehicle_Type.vehicle_type,
    AVG(Buy.sale_date - Vehicle.purchase_date) AS '$average_difference'
FROM
    Vehicle_Type
JOIN
    Vehicle ON Vehicle.vehicle_type = Vehicle_Type.vehicle_type
JOIN
    Buy ON Buy.VIN= Vehicle.VIN
WHERE
    Vehicle.purchase_date IS NOT NULL AND
    Buy.sale_date IS NOT NULL
GROUP BY
    Vehicle_Type.vehicle_type;
```

## Price Per Condition

- Compute the average vehicle price (Vehicle.purchase price) for all vehicles that has the same Vehicle Type.vehicle type and same Vehicle.condition, for all type and all conditions
  - Left Joint the Vehicle Type and Vehicle with vehicle type
  - Group the results by Vehicle Type.vehicle type
  - Calculate the average of Vehicle.purchase price for each Vehicle.condition
- Display Vehicle Type.vehicle type, average purchase price, and Vehicle.condition on **Price per Condition** form.

```
SELECT
    Vehicle_Type.vehicle_type,
```

```
    AVG(CASE WHEN Vehicle.vehicle_condition = 'Excellent' THEN
    Vehicle.purchase_price END) AS 'Excellent',
    AVG(CASE WHEN Vehicle.vehicle_condition = 'Very Good' THEN
    Vehicle.purchase_price END) AS 'Very Good',
    AVG(CASE WHEN Vehicle.vehicle_condition = 'Good' THEN
    Vehicle.purchase_price END) AS 'Good',
    AVG(CASE WHEN Vehicle.vehicle_condition = 'Fair' THEN
    Vehicle.purchase_price END) AS 'Fair',
FROM
    Vehicle_Type
LEFT JOIN
    Vehicle ON Vehicle_Type.vehicle_type = Vehicle.vehicle_type
GROUP BY
    Vehicle_Type.vehicle_type;
```

## Parts Statistics

- Find the associate Part and Vendor from associate and include relationship of Parts Order
- For each Vendor.name
  - Find and calculate the *total quantity* of all related parts by summing Part.quantity
  - Find and calculate the *total dollar amount* of all related parts by summing Part.unit price*Part.quantity
- Sort the results using *total dollar amount*
- For each Vendor
  - Display Vendor.name, *total quantity* , *total dollar amount* on **Parts Statistics** form.

```
SELECT
    Parts_Order.vendor_name,
    SUM(Part.quantity) AS total_quantity,
    SUM(Part.quantity * Part.unit_price) AS total_dollar_amount
FROM  `Parts_Order`
INNER JOIN Part ON Parts_Order.VIN = Part.VIN
                    AND Parts_Order.order_number = Part.order_number
GROUP BY  Part.vendor_name
ORDER BY  total_dollar_amount DESC;
```

## Monthly Sales

- Find and calculate the sum of Parts Order.total cost for each sold Vehicle.VIN as *Total expenses*
- Find the *sold record* from buy relationship for each sold Vehicle.VIN
  - Find the corresponding *Sale month* and *Sale year* using buy.sale date
  - Find the corresponding *Sale price* using Vehicle.sale price
- Group the results for each year and each month:
  - Count the total number of vehicles
  - Sum all Vehicle.sale price as *Gross sale income*
  - Sum all (Vehicle.sale price-Vehicle.purchase price-*Total expenses* of Vehicle) as *Total net income*
- Sort the results ordered by year and month descending

- Display *Sale year*, *Sale month*, total number of vehicles, *Gross sale income*, and *Total net income* on **Monthly Sales** form.
- If clicked on any month, jump to **Monthly Sales Drill-down** task

```
SELECT
    YEAR(Buy.sale_date) AS sale_year,
    MONTH(Buy.sale_date) AS sale_month,
    COUNT(Buy.VIN) AS total_vehicles_sold,
    SUM(Vehicle.sale_price) AS gross_sales_income,
    SUM(Vehicle.sale_price - Vehicle.purchase_price - COALESCE(p.total_expenses,
    0)) AS total_net_income
FROM
    `Buy`
INNER JOIN Vehicle ON Buy.VIN = Vehicle.VIN
LEFT JOIN (
    SELECT VIN, SUM(total_cost) AS total_expenses
    FROM Parts_Order
    GROUP BY VIN
) p ON Vehicle.VIN = p.VIN
GROUP BY
    sale_year, sale_month
ORDER BY
    sale_year DESC, sale_month DESC;
```

## Monthly Sales Drill-down

- Since the user click on certain year and month to pop-up this drill-down, we will have the input of the $Year and $Month
- Find the *sold record* from buy.sale date for each sold Vehicle.VIN and Salepeople in that $Year and $Month
- Group the results for each Salepeople:
  - Count the total number of vehicles as *vehicles sold*
  - Sum all Vehicle.sale price as *total sales*
- Sort the results ordered by vehicles sold and total sales descending
- Find Logged-in User.username by Salepeople.username:
  - Display Logged-in User.name
  - Display vehicles sold of each salespeople
  - Display total sales of each salespeople

```
SELECT
    Logged_in_User.FirstName,
    Logged_in_User.LastName,
    COUNT(Buy.VIN) AS vehicles_sold,
    SUM(Vehicle.sale_price) AS total_sales
FROM
    `Buy`
INNER JOIN Vehicle ON Buy.VIN = Vehicle.VIN
INNER JOIN Logged_in_User ON Buy.salepeople_username = Logged_in_User.username
WHERE
    YEAR(Buy.sale_date) = '$Year'
    AND MONTH(Buy.sale_date) = '$Month'
```

```
GROUP BY
    Logged_in_User.username
ORDER BY
    vehicles_sold DESC, total_sales DESC;
```