# Table of Contents

# Abstract Code

## Search Vehicle

**Abstract Code**

- User clicks on ***Search Vehicle*** button, jump to search page, where initially display total number of vehicles available for purchase.
- User input *Vehicle type* ($VehicleType)*, Manufacturer*($ManufacturerName)*, Model year*($ModelYear)*, Color*($ColorName)*, List price*($InvoicePrice*1.25) *, Keyword*($Keyword)*.* Privileged user can search by VIN
- User clicks ***Search Vehicle*** button
    - Return unsold vehicles matching all search terms sorted by VIN in ascending order, where display
        - VIN
        - Vehicle type
        - Model year
        - Manufacturer
        - Model name
        - Color
        - If an entered keyword matched the description, indicate this with a mark
        - List price
    - The query's where clause will be dependent on the search terms selected:
        - If VIN is selected, Where clause looks like Vehicle.VIN='$VIN' AND DateSold IS NULL
        - If List Price and Model is selected, Where clause looks like InvoicePrice*1.25<='$ListPrice' and ModelName='$ModelName' AND DateSold IS NULL
    - If the user type equals manager or owner, then they can search for sold vehicles and the WHERE clause would be changed to DateSold IS NOT NULL or removed completely if searching for both types.
    - If  no vehicles meet the criteria, display "Sorry, it looks like we don't have that in stock!"
    - User click any columns header to sort result by the column
    - User click an individual result to open a detail page

Sample Query, but the Where clause will be constructed based on the search options as described above.

```
SELECT Vehicle.VIN,
(CASE
  WHEN DoorCount IS NOT NULL THEN 'Car'
  WHEN CargoCapacity IS NOT NULL THEN 'Truck'
  WHEN RoofType IS NOT NULL THEN 'SUV'
  WHEN DriveTrainType IS NOT NULL THEN 'Convertible'
  WHEN DriverSideDoor IS NOT NULL THEN 'Van'
  END) AS VehicleType,
Vehicle.ModelYear, Vehicle.ManufacturerName, Vehicle.ModelName,
GROUP_CONCAT(ColorName) AS Colors, Vehicle.InvoicePrice*1.25 AS ListPrice,
Vehicle.Description
FROM Vehicle
JOIN Vehicle_Color ON Vehicle.VIN = Vehicle_Color.VIN
LEFT JOIN Vehicle_Car ON Vehicle.VIN = Vehicle_Car.VIN
LEFT JOIN Vehicle_Truck ON Vehicle.VIN = Vehicle_Truck .VIN
LEFT JOIN Vehicle_SUV ON Vehicle.VIN = Vehicle_SUV.VIN
LEFT JOIN Vehicle_Convertible ON Vehicle.VIN = Vehicle_Convertible.VIN
LEFT JOIN Vehicle_Van ON Vehicle.VIN = Vehicle_Van.VIN
WHERE (ManufacturerName='$ManufacturerName' AND ModelYear='$ModelYear' AND
ColorName='$ColorName' AND InvoicePrice*1.25<='$ListPrice' AND Description LIKE
'%$Keyword%' AND DateSold IS NULL)
OR Vehicle.VIN='$VIN'
GROUP BY Vehicle.VIN
ORDER BY Vehicle.VIN ASC;
```

# Login

**Abstract Code**

- User click ***Login*** on the ***Main Menu***
- User enters *username*($Username) and *password*($Password)
    - If validation for both username and password is correct, then page will update to include access to the appropriate functionality based on the user duty.

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

        - If User record is found but User.Password != '$Password'
            - Go back to ***Login*** form, with error message
        - Else:
            - Store login information as session variable '$Username'.

## Add Vehicle

**Abstract Code:**

- After login, User click ***Add Vehicle*** to load new vehicle form
- User enters vehicle details, *VIN* ($VIN), vehicle type ($VehicleType) with any specific attributes, ManufacturerName ($ManufacturerName), ModelName ($ModelName), ModelYear ($ModelYear), List of Colors ([$ColorName]), invoice price ($InvoicePrice), Description($Description), *clerk username* ($ClerkUserName) and *date added* ($DateAdded).
- User clicks ***submit*** button.
    - Write to *Vehicle*

```
-- insert into table Vehicle
INSERT INTO Vehicle(VIN, ManufacturerName, ModelName, ModelYear, InvoicePrice,
Description, ClerkUserName, DateAdded)
VALUES ('$VIN', '$ManufacturerName', '$ModelName', '$ModelYear',
'$InvoicePrice', '$Description', '$ClerkUserName', '$DateAdded');
```

    - Write to *Vehicle_Color* (for each $ColorName in the list)

```
-- insert into table Vehicle_Color
INSERT INTO Vehicle_Color(VIN, ColorName)
Values ('$VIN', '$ColorName');
```

    - Write to *Vehicle_<$VehicleType>* UI handles $VehicleType and run proper query

```
-- insert into vehicle subclass table
INSERT INTO Vehicle_Car(VIN, DoorCount) VALUES ('$VIN', '$DoorCount')

INSERT INTO Vehicle_Truck(VIN, CargoCapacity, CargoCoverType, RearAxisCount)
VALUES ('$VIN', '$CargoCapacity', '$CargoCoverType', '$RearAxisCount')

INSERT INTO Vehicle_SUV(VIN, RoofType, BackSeatCount) VALUES ('$VIN',
'$RoofType', '$BackSeatCount')

INSERT INTO Vehicle_Convertible(VIN, DriveTrainType, CupholderCount) VALUES
('$VIN', '$DriveTrainType', '$CupHolderCount')

INSERT INTO Vehicle_Van(VIN, DriverSideDoor) VALUES ('$VIN',
'$DriverSideDoor')
```

    - Display vehicle detail page
    - Vehicle is available for sale

## Sell Vehicle

**Abstract Code**

- Run ***Search vehicle*** task to check available vehicles
- click a vehicle to view details.
- User clicks the ***Sell vehicle*** button to sell the vehicle
    - Load the sales order form
    - Find/Add customer ($CustomerID) - *note: SQL located at Add/Find Customer*
        - Look up a customer
        - Add a customer if not exist
    - Confirm the sale by entering the sold price ( $SoldPrice) and sold date ($DateSold)

```
UPDATE Vehicle
SET CustomerID = '$CustomerID', SoldPrice = '$SoldPrice', DateSold =
'$DateSold', SalespeopleUsername =  '$Username'
WHERE VIN = '$VIN';
```

# View Vehicle

**Abstract Code**

- User clicks a vehicle link to open a detailed page ($VIN)
- Display VIN, vehicle type, attributes for that vehicle type, Model Year, Model Name, Manufacturer, color(s), list price and the description.
- If Usertype is NOT manager or owner, then only the anonymous query is fired off to retrieve the vehicle information
- If the UserType is manager or owner, then additional inventory addition query, sales query and the respair query are fired off to retrieve additional information that the two UserType can see on the vehicle details page.

```
-- Anonymous access
SELECT Vehicle.VIN, Vehicle.ModelYear, Vehicle.ModelName, Vehicle.ManufacturerName,
(CASE
  WHEN DoorCount IS NOT NULL THEN 'Car'
  WHEN CargoCapacity IS NOT NULL THEN 'Truck'
  WHEN RoofType IS NOT NULL THEN 'SUV'
  WHEN DriveTrainType IS NOT NULL THEN 'Convertible'
  WHEN DriverSideDoor IS NOT NULL THEN 'Van'
  END) AS VehicleType,
GROUP_CONCAT(ColorName) AS Colors, Vehicle_Car.DoorCount, Vehicle_Truck.
CargoCapacity, Vehicle_Truck. CargoCoverType, Vehicle_Truck. RearAxisCount,
Vehicle_SUV.RoofType, Vehicle_SUV.BackSeatCount,
Vehicle_Convertible.DriveTrainType, Vehicle_Convertible.CupHolderCount,
Vehicle_Van. DriverSideDoor
FROM Vehicle
JOIN Vehicle_Color ON Vehicle.VIN = Vehicle_Color.VIN
LEFT JOIN Vehicle_Car ON Vehicle.VIN = Vehicle_Car.VIN
LEFT JOIN Vehicle_Truck ON Vehicle.VIN = Vehicle_Truck .VIN
LEFT JOIN Vehicle_SUV ON Vehicle.VIN = Vehicle_SUV.VIN
LEFT JOIN Vehicle_Convertible ON Vehicle.VIN = Vehicle_Convertible.VIN
LEFT JOIN Vehicle_Van ON Vehicle.VIN = Vehicle_Van.VIN
WHERE Vehicle.VIN = '$VIN';

-- Manager/Owner access
-- Inventory Add query
SELECT CONCAT(User.FirstName, ' ', User.LastName) AS ClerkName, InvoicePrice,
DateAdded
FROM Vehicle
LEFT JOIN User ON Vehicle.ClerkUsername = User.Username
WHERE Vehicle.VIN = '$VIN';

-- Sales Query (buyer info will only be displayed if the vehicle is sold)
```

```
SELECT CONCAT(User.FirstName, ' ', User.LastName) AS SalespersonName,
Email, StreetAddress, City, State, ZipCode, Phone, I.FirstName, I.LastName,
BusinessName, PrimaryContactName, PrimaryContactTitle,
DateSold, SoldPrice, InvoicePrice, DateAdded
FROM Vehicle
JOIN User ON Vehicle.SalespeopleUsername = User.Username
JOIN Customer ON Vehicle.CustomerID = Customer.CustomerID
LEFT JOIN Individual AS I ON Vehicle.CustomerID = I.CustomerID
LEFT JOIN Business AS B ON Vehicle.CustomerID = B.CustomerID
WHERE Vehicle.VIN = '$VIN' AND DateSold IS NULL;

-- Repair Query
SELECT Repair.StartDate, EndDate, LaborCharges, COALESCE(UnitPrice*QuantityUsed, 0)
AS PartsCost, (COALESCE(UnitPrice*QuantityUsed, 0) + LaborCharges) AS TotalCost,
CONCAT(User.FirstName, ' ', User.LastName) AS ServiceWriterName,
COALESCE(CONCAT(Individual.FirstName, ' ',Individual.LastName), BusinessName) AS
CustomerName
FROM Repair
INNER JOIN User ON Repair.ServiceWriterUsername = User.Username
INNER JOIN Vehicle ON Repair.VIN = Vehicle.VIN
LEFT JOIN Part ON Repair.VIN = Part.VIN AND Repair.StartDate = Part.StartDate
INNER JOIN Customer ON Repair.CustomerID = Customer.CustomerID
LEFT JOIN Individual ON Repair.CustomerID = Individual.CustomerID
LEFT JOIN Business ON Repair.ServiceWriterUsername = Business .CustomerID
WHERE Repair.VIN = '$VIN';
```

# View Repair Form

## Abstract Code

- Service Writer clicks on ***Open Repair Form*** link / button
- Service Writer will be prompted to enter a VIN ($VIN)

```
SELECT Vehicle.VIN, ModelYear,ModelName, DateSold,Vehicle.ManufacturerName,
(CASE
  WHEN DoorCount IS NOT NULL THEN 'Car'
  WHEN CargoCapacity IS NOT NULL THEN 'Truck'
  WHEN RoofType IS NOT NULL THEN 'SUV'
  WHEN DriveTrainType IS NOT NULL THEN 'Convertible'
  WHEN DriverSideDoor IS NOT NULL THEN 'Van'
  END) AS VehicleType,
GROUP_CONCAT(ColorName) AS Colors
FROM Vehicle
JOIN Vehicle_Color ON Vehicle.VIN = Vehicle_Color.VIN
LEFT JOIN Vehicle_Car ON Vehicle.VIN = Vehicle_Car.VIN
LEFT JOIN Vehicle_Truck ON Vehicle.VIN = Vehicle_Truck .VIN
LEFT JOIN Vehicle_SUV ON Vehicle.VIN = Vehicle_SUV.VIN
LEFT JOIN Vehicle_Convertible ON Vehicle.VIN = Vehicle_Convertible.VIN
LEFT JOIN Vehicle_Van ON Vehicle.VIN = Vehicle_Van.VIN
WHERE Vehicle.VIN='$VIN';
```

- If no such vehicle has been found:
    - An appropriate error message will be displayed
- If vehicle record is found but Vehicle.`DateSold` is NULL:
    - An appropriate error message will be displayed
- Else:
    - The repair form will show following fields: VIN, vehicle type, Model year, Model Name, Manufacturer and color(s)

# Update Repair Form

## Abstract Code

- Service Writer can open the **Repair Form** and view vehicle details
- The application will look for this vehicle in Repair table

```
SELECT VIN, StartDate, LaborCharges, EndDate FROM Repair WHERE
Repair.VIN='$VIN';
```

- If record(s) is found and Repair.EndDate IS NULL
    - Service Writer is allowed to update **Repair Form** by entering labor charges($LaborCharges) or add parts($QuantityUsed, $VendorName, $PartNumber, $UnitPrice) or complete repair
        - If $LaborCharges is less than Repair.LaborCharges, display error message
        - Otherwise
        - $EndDate is today's date

```
INSERT INTO Part(VIN, StartDate, PartNumber, QuantityUsed, UnitPrice,
VendorName) VALUES ('$VIN', '$StartDate', '$PartNumber', '$QuantityUsed',
'$UnitPrice', '$VendorName');

UPDATE Repair SET LaborCharges='$LaborCharges', Description='$Description',
EndDate='$EndDate' WHERE VIN='$VIN' AND StartDate='$StartDate';
```

# Create a New Repair

## Abstract Code

- Service Writer can open the **Repair Form** and view vehicle details
- The application will look for this vehicle in Repair table (we check whether an incomplete repair exist on the vehicle)

```
SELECT VIN
FROM Repair
WHERE Repair.VIN='$VIN' AND Repair.EndDate IS NULL;
```

- If no record (aka no incomplete repair) has been found:
    - Service Writer is allowed to start a new repair by entering
        - Odometer reading ($Odometer)
        - Associate an existing Customer or Add a new Customer and take CustomerID by **Add/Find Customer**($CustomerID)
        - $StartDate is today's date
        - Description of the repair ($Description)

```
INSERT INTO Repair(VIN, StartDate, CustomerID, Odometer, Description,
ServiceWriterUsername) VALUES ('$VIN', '$StartDate', '$CustomerID',
'$Odometer', '$Description', '$Username');
```

    - Service writer will be allowed to enter labor charges($LaborCharges) and add parts($QuantityUsed, $VendorName, $PartNumber, $UnitPrice) after repair is created
        - If data validation is successful for all parts input
            - $QuantityUsed should be integer
            - $PartNumber should be integer
            - $UnitPrice should be float number
        - $StartDate is today's date

```
INSERT INTO Part(VIN, StartDate, PartNumber, QuantityUsed, UnitPrice,
VendorName) VALUES ('$VIN', '$StartDate', '$PartNumber', '$QuantityUsed',
'$UnitPrice', '$VendorName');

UPDATE Repair SET LaborCharges='$LaborCharges', Description='$Description'
WHERE VIN='$VIN' AND StartDate='$StartDate';
```

        - If input fields are invalid, display error messages

## Add/Find Customer

### Abstract Code

- If the signed in User's Usertype equals ServiceWriter, Salespeople or Owner, then they can look up a customer in their flow with either TaxID or DriverLicense entered in a single field (`$inputID`)
    - Two queries are fired off.

```
SELECT CustomerId, BusinessName
FROM Business
WHERE Business.TaxID = '$inputID';

SELECT CustomerId, CONCAT(FirstName, ' ', LastName) as
IndividualName
FROM Individual
WHERE Individual.DriverLicense = '$inputID';
```

- If the customer exists then one of the queries will return a result and we will populate that result back in the sales or the repair page where the user is adding the customer.
- If no result is found, then:
    - Option to add new customer is provided
- User choose a Customer type, dropdown between Individual or Business, to add new Customer
- No matter what is selected:
    - User needs to enter
        - *Phone number* (must be formatted either as ten digit or xxx-xxx-xxxx) (`$phone`)
        - *Street Address* (`$streetAddress`)
        - *City* (`$city`)
        - *State* (`$state`)
        - *Zipcode* (`$zipcode`)
        - *Email* is optional (`$email`)
    - Query to insert into Customer table (used for both Individual and Business scenario) with variables from the form.

```
INSERT INTO Customer (Email, StreetAddress, City, State, ZipCode,
Phone)
VALUES ('$email', '$streetAddress', '$city', '$state', '$zipcode',
'$phone');
```

    - Upon inserting into the Customer table, a CustomerID will be automatically generated by the table because we set that field to AUTO_INCREMENT.
    - We need to then retrieve this CustomerID (`$customerID`) for inserting into the Individual or Business table. We make sure the customerID we are fetching has no match in the Individual and Business table.

```
SELECT Customer.CustomerID
FROM Customer
LEFT JOIN Individual ON Customer.CustomerID = Individual.CustomerID
LEFT JOIN Business ON Customer.CustomerID = Business.CustomerID
```

```
WHERE StreetAddress='$streetAddress' AND City='$city' AND
State='$state' AND ZipCode='$zipcode' AND Phone='$phone' AND
Business.TaxID IS NULL AND Individual.DriverLicense IS NULL
```

- Alternative way of fetching CustomerID based on the assuming that there is only one User logged in at a time based on piazza post of 932. We can just assume the last added customerID is correct customerID to insert into Individual or Business.

```
SELECT CustomerID
FROM Customer
ORDER BY CustomerID DESC LIMIT 1;
```

- If an individual is selected, then:
  - User needs to enter
    - First (`$firstName`) and Last Name (`$lastName`)
    - Driver's license number (`$driverLicense`)
  - Query to insert into Individual table with variables from the form.

```
INSERT INTO Individual (DriverLicense, CustomerID, FirstName,
LastName)
VALUES ('$driverLicense', '$customerID', '$firstName', '$lastName');
```

  - If the insert fails with a duplicate key error then User will be asked to update the DriverLicense
- If a business is to be added, then:
  - User needs to record
    - Business Tax Identification Id (`$taxID`)
    - Business Name (`$businessName`)
    - Name of Primary Contact (`$primaryContactName`)
    - Contact Title (`$primaryContactTitle`)
  - Query to insert into Business with variables from the form.

```
INSERT INTO Business (TaxID, CustomerID, BusinessName,
PrimaryContactName, PrimaryContactTitle)
VALUES ('$taxID', '$customerID', '$businessName',
'$primaryContactName', '$primaryContactTitle');
```

  - If the insert fails with a duplicate key error then User will be asked to update the TaxID

## Open Report

### Abstract Code

- Manager or Owner clicks the “Report” button to run the Open ***Report*** task.
- From the expanded menu, click the report type to generate, the report will be displayed on a new page using the query specified for each report type..
- Case “Sales by Color”
    - Read Color table, left join with Vehicle_Color and Vehicle
    - Grouped by *Color*
    - Generate three temporary tables in the process to calculate sales for each “within 30 days”, “within a year”, “any”.
    - Combine results and return. If a certain color does not have any sales, query will return 0

```
SELECT tmp1.ColorName, COALESCE(ALLTIME,0), COALESCE(YEARCOUNT,0),
COALESCE(DAYCOUNT,0) FROM
(
  SELECT ColorName, COUNT(VIN) AS ALLTIME
  FROM Color
  LEFT JOIN
  (
    SELECT Vehicle.VIN,(CASE
    WHEN GROUP_CONCAT(Vehicle_Color.ColorName) LIKE '%,%' THEN
'multiple'
    ELSE GROUP_CONCAT(Vehicle_Color.ColorName)
    END) AS ColorType, DateSold
    FROM Vehicle
    LEFT JOIN Vehicle_Color ON Vehicle.VIN = Vehicle_Color.VIN
    WHERE Vehicle.CustomerID IS NOT NULL
    GROUP BY Vehicle.VIN
  ) colormulti1 ON Color.ColorName = colormulti1.ColorType
  GROUP BY ColorName
) tmp1
LEFT JOIN
(
  SELECT ColorName, COUNT(VIN) AS YEARCOUNT
  FROM Color
  LEFT JOIN
  (SELECT Vehicle.VIN,(CASE
    WHEN GROUP_CONCAT(Vehicle_Color.ColorName) LIKE '%,%' THEN
'multiple'
    ELSE GROUP_CONCAT(Vehicle_Color.ColorName)
    END) AS ColorType, DateSold
```

```
    FROM Vehicle
    LEFT JOIN Vehicle_Color ON Vehicle.VIN = Vehicle_Color.VIN
    WHERE Vehicle.CustomerID IS NOT NULL
    GROUP BY Vehicle.VIN) colormulti2 ON Color.ColorName =
colormulti2.ColorType
  WHERE DATEDIFF(NOW(), DateSold-1) < 365
  GROUP BY ColorName
) tmp2  ON tmp1.ColorName = tmp2.ColorName
LEFT JOIN
(
  SELECT ColorName, COUNT(VIN) AS DAYCOUNT
  FROM Color
  LEFT JOIN
  (SELECT Vehicle.VIN,(CASE
    WHEN GROUP_CONCAT(Vehicle_Color.ColorName) LIKE '%,%' THEN
'multiple'
    ELSE GROUP_CONCAT(Vehicle_Color.ColorName)
    END) AS ColorType, DateSold
    FROM Vehicle
    LEFT JOIN Vehicle_Color ON Vehicle.VIN = Vehicle_Color.VIN
    WHERE Vehicle.CustomerID IS NOT NULL
    GROUP BY Vehicle.VIN) colormulti3 ON Color.ColorName =
colormulti3.ColorType
  WHERE DATEDIFF(NOW(), DateSold-1) < 30
  GROUP BY ColorName
) tmp3 ON tmp1.ColorName = tmp3.ColorName;
```

    - Display the form where colors are the rows. where ALLTIME matches up with “any”, YEARCOUNT matches up with “within a year” and DAYCOUNT matches up with “within 30 days”.
- Case “Sales by Type”
    - Read all *Vehicle* entities
    - Break into vehicle types based on mapping to the subclass table.
    - Generate three temporary tables in the process to calculate sales for each “within 30 days”, “within a year”, “any”.
    - Combine results and return. If a certain type does not have any sales, query will return 0

```
SELECT tmp1.VehicleType, COALESCE(ALLTIME,0), COALESCE(YEARCOUNT,0),
COALESCE(DAYCOUNT,0) FROM
(SELECT
(CASE
```

```
  WHEN DoorCount IS NOT NULL THEN 'Car'
  WHEN CargoCapacity IS NOT NULL THEN 'Truck'
  WHEN RoofType IS NOT NULL THEN 'SUV'
  WHEN DriveTrainType IS NOT NULL THEN 'Convertible'
  WHEN DriverSideDoor IS NOT NULL THEN 'Van'
  END) AS VehicleType, COUNT(v.VIN) as ALLTIME
FROM Vehicle v
LEFT JOIN Vehicle_Car ON v.VIN = Vehicle_Car.VIN
LEFT JOIN Vehicle_Truck ON v.VIN = Vehicle_Truck .VIN
LEFT JOIN Vehicle_SUV ON v.VIN = Vehicle_SUV.VIN
LEFT JOIN Vehicle_Convertible ON v.VIN = Vehicle_Convertible.VIN
LEFT JOIN Vehicle_Van ON v.VIN = Vehicle_Van.VIN
WHERE v.CustomerID IS NOT NULL
GROUP BY VehicleType
) tmp1
LEFT JOIN
(SELECT
(CASE
  WHEN DoorCount IS NOT NULL THEN 'Car'
  WHEN CargoCapacity IS NOT NULL THEN 'Truck'
  WHEN RoofType IS NOT NULL THEN 'SUV'
  WHEN DriveTrainType IS NOT NULL THEN 'Convertible'
  WHEN DriverSideDoor IS NOT NULL THEN 'Van'
  END) AS VehicleType, COUNT(v.VIN) as YEARCOUNT
FROM Vehicle v
LEFT JOIN Vehicle_Car ON v.VIN = Vehicle_Car.VIN
LEFT JOIN Vehicle_Truck ON v.VIN = Vehicle_Truck .VIN
LEFT JOIN Vehicle_SUV ON v.VIN = Vehicle_SUV.VIN
LEFT JOIN Vehicle_Convertible ON v.VIN = Vehicle_Convertible.VIN
LEFT JOIN Vehicle_Van ON v.VIN = Vehicle_Van.VIN
WHERE v.CustomerID IS NOT NULL AND DATEDIFF(NOW(), DateSold-1) < 365
GROUP BY VehicleType) tmp2 ON tmp1.VehicleType = tmp2.VehicleType
LEFT JOIN
(SELECT
(CASE
  WHEN DoorCount IS NOT NULL THEN 'Car'
  WHEN CargoCapacity IS NOT NULL THEN 'Truck'
  WHEN RoofType IS NOT NULL THEN 'SUV'
  WHEN DriveTrainType IS NOT NULL THEN 'Convertible'
  WHEN DriverSideDoor IS NOT NULL THEN 'Van'
  END) AS VehicleType, COUNT(v.VIN) as DAYCOUNT
```

```
FROM Vehicle v
LEFT JOIN Vehicle_Car ON v.VIN = Vehicle_Car.VIN
LEFT JOIN Vehicle_Truck ON v.VIN = Vehicle_Truck .VIN
LEFT JOIN Vehicle_SUV ON v.VIN = Vehicle_SUV.VIN
LEFT JOIN Vehicle_Convertible ON v.VIN = Vehicle_Convertible.VIN
LEFT JOIN Vehicle_Van ON v.VIN = Vehicle_Van.VIN
WHERE v.CustomerID IS NOT NULL AND DATEDIFF(NOW(), DateSold-1) < 30
GROUP BY VehicleType
) tmp3 ON tmp1.VehicleType = tmp3.VehicleType;
```

    - Display the form where types are the rows. where ALLTIME matches up with “any”, YEARCOUNT matches up with “within a year” and DAYCOUNT matches up with “within 30 days”.
- Case “Sales by Manufacturer”
    - Read all *Vehicle* entities
    - Grouped by *Manufacturer*
    - Generate three temporary tables in the process to calculate sales for each “within 30 days”, “within a year”, “any”.
    - Combine results and return.

```
SELECT tmp1.ManufacturerName, COALESCE(ALLTIME,0),
COALESCE(YEARCOUNT,0), COALESCE(DAYCOUNT,0) FROM
  (SELECT ManufacturerName, COUNT(Vehicle.VIN) as ALLTIME
  FROM Vehicle
  WHERE Vehicle.CustomerID IS NOT NULL
  GROUP BY ManufacturerName
  ) tmp1
LEFT JOIN
  (SELECT ManufacturerName, COUNT(Vehicle.VIN) as YEARCOUNT
  FROM Vehicle
  WHERE Vehicle.CustomerID IS NOT NULL AND DATEDIFF(NOW(),
DateSold-1) < 365
  GROUP BY ManufacturerName) tmp2 ON tmp1.ManufacturerName =
tmp2.ManufacturerName
LEFT JOIN
  (SELECT ManufacturerName, COUNT(Vehicle.VIN) as DAYCOUNT
  FROM Vehicle
  WHERE Vehicle.CustomerID IS NOT NULL AND DATEDIFF(NOW(),
DateSold-1) < 30
  GROUP BY ManufacturerName) tmp3 ON tmp1.ManufacturerName =
tmp3.ManufacturerName;
```

    - Display the form where manufacturers are the rows, where ALLTIME matches up with “any”, YEARCOUNT matches up with “within a year” and DAYCOUNT matches up with “within 30 days”.
- Case “Gross Customer Income”
    - Select top 15 customers with following attributes: name, the date of the first sale or repair start date, the date of their most recent sale or repair start date, their number of sales, their number of repairs, and the gross income using the results from the steps below.
        - Read all sold *Vehicle* entities, joined with *User*, find the salespeople’s name
        - Grouped by *CustomerID*, sum the *SoldPrice*. Keep all records for each customer in a table.
        - Read all *Repair* entities, joined with *Part*, compute the repair cost
        - Grouped by *CustomerID*, sum the total repair cost from each record. Keep all records for each customer in a table.
        - Sum up the 2 types of incomes from each customer and keep the vehicle purchase records with repair records together, ordered by total income descending and then by last sale/repair start date descending.
        - Read customer’s name (first/last for individuals or business name for business) from *Customer* entities.

```
-- report part 1
SELECT
s.CustomerID,
COALESCE(CONCAT(i.FirstName, ' ',i.LastName), BusinessName) AS CustomerName,
LEAST(FirstSalesDate,FirstRepairDate) AS FirstTransactionDate,
GREATEST(LastSalesDate,LastRepairDate) AS LastTransactionDate, NumberOfSales,
NumberOfRepairs,
(TotalSalesIncome+TotalRepairIncome) AS TotalTransaction
FROM (SELECT CustomerID,
      SUM(SoldPrice) AS TotalSalesIncome,
      COUNT(VIN) AS NumberOfSales,
      MIN(DateSold) AS FirstSalesDate,
      MAX(DateSold) AS LastSalesDate
      FROM (SELECT CustomerID, DateSold, SoldPrice, VIN, ModelYear, ManufacturerName,
ModelName,
            CONCAT(u.FirstName,' ',u.LastName) AS SalespeopleName
            FROM Vehicle v
            JOIN User u ON v.SalespeopleUsername = u.Username
            WHERE CustomerID IS NOT NULL) sales_income
      GROUP BY CustomerID) s
JOIN (SELECT CustomerID,
      SUM(LaborCharges+PartsCost) AS TotalRepairIncome,
      COUNT(VIN) AS NumberOfRepairs,
      MIN(StartDate) AS FirstRepairDate,
```

```
      MAX(StartDate) AS LastRepairDate
      FROM (SELECT CustomerID, r.StartDate, EndDate, r.VIN, Odometer, LaborCharges,
            COALESCE((UnitPrice*QuantityUsed), 0)
            AS PartsCost, CONCAT(u.FirstName,' ',u.LastName) AS ServiceWriterName
            FROM Repair r
            LEFT JOIN Part p ON r.VIN = p.VIN AND r.StartDate = p.StartDate
            JOIN User u ON r.ServiceWriterUsername = u.Username) repair_income
      GROUP BY CustomerID) r ON s.CustomerID = r.CustomerID
JOIN Individual i ON s.CustomerID = i.CustomerID
JOIN Business ON s.CustomerID = Business.CustomerID
ORDER BY TotalTransaction, LastTransactionDate DESC
LIMIT 15;
```

- Display the form of the top 15 customers
- Click on the link within any customer's name (passing CustomerID to drill-down)
- Expand the 2 lists of records stored priorly into:
  - Vehicle Sales Section: sale date, sold price, VIN, year, manufacturer, model, and salesperson name (Read from *User* entities given *SalespeopleUsername*), for each sale, sorted by sale date descending and VIN ascending.

```
-- drill-down vehicle section
SELECT DateSold, SoldPrice, VIN, ModelYear, ManufacturerName, ModelName,
SalespeopleName
FROM (SELECT CustomerID, DateSold, SoldPrice, VIN, ModelYear,
ManufacturerName, ModelName,
      CONCAT(u.FirstName,' ',u.LastName) AS SalespeopleName
      FROM Vehicle v
      JOIN User u ON v.SalespeopleUsername = u.Username
      WHERE CustomerID IS NOT NULL) sales_income
WHERE CustomerID = '$CustomerID'
ORDER BY DateSold DESC, VIN;
```

  - Repairs Section: start date, end date (if the repair is not finished, this should not display any value), the VIN of the repaired vehicle, the odometer reading, parts cost, labor cost, total cost, and the service writer who opened the repair, for each repair, sorted by start date descending, end date descending, and VIN ascending; however, any incomplete repairs should be listed before completed ones with the same sorting criteria.

```
-- drill-down repair section
SELECT StartDate, EndDate, VIN, Odometer, PartsCost, LaborCharges,
(PartsCost+LaborCharges) AS TotalCost, ServiceWriterName
```

```
FROM (SELECT CustomerID, r.StartDate, EndDate, r.VIN, Odometer, LaborCharges,
      COALESCE((UnitPrice*QuantityUsed), 0)
      AS PartsCost, CONCAT(u.FirstName,' ',u.LastName) AS ServiceWriterName
      FROM Repair r
      LEFT JOIN Part p ON r.VIN = p.VIN AND r.StartDate = p.StartDate
      JOIN User u ON r.ServiceWriterUsername = u.Username) repair_income
WHERE CustomerID = '$CustomerID'
ORDER BY StartDate DESC, EndDate IS NOT NULL, EndDate DESC, VIN;
```

    - Display the drilldown report with 2 sections

- Case “Repairs by Manufacturer/Type/Model”
    - Read and join all *Vehicle, Repair, Part* entities.
    - Keeping all *Manufacturer,* grouped by *Manufacturer*, compute the count of repairs, the sum of all parts costs, the sum of all labor costs, and the sum of total repair costs, including any repairs in progress. Keep the list of repairs for each manufacturer, sorted by manufacturer name ascending (first part).

```
-- first part
SELECT
ManufacturerName,
SUM(StartDate IS NOT NULL) AS RepairCount,
SUM(PartsCost) AS TotalPartsCost,
SUM(LaborCharges) AS TotalLaborCharges,
SUM(RepairCost) AS TotalRepairCost
FROM (SELECT v.VIN, ModelName, ManufacturerName, StartDate,
      COALESCE(LaborCharges, 0) AS LaborCharges,
      COALESCE(PartsCost, 0) AS PartsCost,
      COALESCE(LaborCharges+PartsCost, 0) AS RepairCost
      FROM Vehicle v
      LEFT JOIN
      (SELECT r.VIN, r.StartDate, r.LaborCharges,
       COALESCE(UnitPrice*QuantityUsed, 0) AS PartsCost
       FROM Repair r
       LEFT JOIN Part p ON r.VIN = p.VIN AND r.StartDate =
p.StartDate) rp
       ON v.VIN = rp.VIN) r
GROUP BY ManufacturerName
ORDER BY ManufacturerName;
```

    - Click on the link within any manufacturer’s name
    - Resolving vehicle types and join *VehicleType* column, filtered with a given manufacturer, grouped by *Type*, compute the count of repairs, the sum of all parts costs, the sum of all

labor costs, and the sum of total repair costs, including any repairs in progress. If the result is empty, exclude the vehicle type. Ordered by decreasing repair count.

- Resolving vehicle types, filtered with the given manufacturer from the last step, grouped by *Model*, compute the count of repairs, the sum of all parts costs, the sum of all labor costs, and the sum of total repair costs, including any repairs in progress. Mark the list generated here as “details”. Ordered by vehicle type (for easier separation) then decreasing repair count.

```
-- drill-down part
-- repair by type
SELECT t.VehicleType,
COUNT(StartDate) AS RepairCount,
SUM(PartsCost) AS TotalPartsCost,
SUM(LaborCharges) AS TotalLaborCharges,
SUM(RepairCost) AS TotalRepairCost
FROM (SELECT v.VIN, ModelName, ManufacturerName, StartDate,
      COALESCE(LaborCharges, 0) AS LaborCharges,
      COALESCE(PartsCost, 0) AS PartsCost,
      COALESCE(LaborCharges+PartsCost, 0) AS RepairCost
      FROM Vehicle v
      LEFT JOIN
      (SELECT r.VIN, r.StartDate, r.LaborCharges,
       COALESCE(UnitPrice*QuantityUsed, 0) AS PartsCost
       FROM Repair r
       LEFT JOIN Part p ON r.VIN = p.VIN AND r.StartDate = p.StartDate) rp
       ON v.VIN = rp.VIN) r
JOIN (SELECT v.VIN,
      (CASE
        WHEN DoorCount IS NOT NULL THEN 'Car'
        WHEN CargoCapacity IS NOT NULL THEN 'Truck'
        WHEN RoofType IS NOT NULL THEN 'SUV'
        WHEN DriveTrainType IS NOT NULL THEN 'Convertible'
        WHEN DriverSideDoor IS NOT NULL THEN 'Van'
        END) AS VehicleType
      FROM Vehicle v
      LEFT JOIN Vehicle_Car ON v.VIN = Vehicle_Car.VIN
      LEFT JOIN Vehicle_Truck ON v.VIN = Vehicle_Truck .VIN
      LEFT JOIN Vehicle_SUV ON v.VIN = Vehicle_SUV.VIN
      LEFT JOIN Vehicle_Convertible ON v.VIN = Vehicle_Convertible.VIN
      LEFT JOIN Vehicle_Van ON v.VIN = Vehicle_Van.VIN) t ON r.VIN = t.VIN
WHERE ManufacturerName = '$ManufacturerName'
AND StartDate IS NOT NULL
```

```
GROUP BY t.VehicleType
ORDER BY RepairCount DESC;

-- repair by model (detail rows for all models)
SELECT ModelName, VehicleType,
COUNT(StartDate) AS RepairCount,
SUM(PartsCost) AS TotalPartsCost,
SUM(LaborCharges) AS TotalLaborCharges,
SUM(RepairCost) AS TotalRepairCost
FROM (SELECT v.VIN, ModelName, ManufacturerName, StartDate,
      COALESCE(LaborCharges, 0) AS LaborCharges,
      COALESCE(PartsCost, 0) AS PartsCost,
      COALESCE(LaborCharges+PartsCost, 0) AS RepairCost
      FROM Vehicle v
      LEFT JOIN
      (SELECT r.VIN, r.StartDate, r.LaborCharges,
       COALESCE(UnitPrice*QuantityUsed, 0) AS PartsCost
       FROM Repair r
       LEFT JOIN Part p ON r.VIN = p.VIN AND r.StartDate = p.StartDate) rp
       ON v.VIN = rp.VIN) r
JOIN (SELECT v.VIN,
      (CASE
        WHEN DoorCount IS NOT NULL THEN 'Car'
        WHEN CargoCapacity IS NOT NULL THEN 'Truck'
        WHEN RoofType IS NOT NULL THEN 'SUV'
        WHEN DriveTrainType IS NOT NULL THEN 'Convertible'
        WHEN DriverSideDoor IS NOT NULL THEN 'Van'
        END) AS VehicleType
      FROM Vehicle v
      LEFT JOIN Vehicle_Car ON v.VIN = Vehicle_Car.VIN
      LEFT JOIN Vehicle_Truck ON v.VIN = Vehicle_Truck .VIN
      LEFT JOIN Vehicle_SUV ON v.VIN = Vehicle_SUV.VIN
      LEFT JOIN Vehicle_Convertible ON v.VIN = Vehicle_Convertible.VIN
      LEFT JOIN Vehicle_Van ON v.VIN = Vehicle_Van.VIN) t ON r.VIN = t.VIN
WHERE ManufacturerName = '$ManufacturerName'
AND StartDate IS NOT NULL
GROUP BY ModelName, VehicleType
ORDER BY VehicleType, RepairCount DESC;
-- Ordered by VehicleType first to make it easy to separate and distribute
under each type.
```

    - Display the drill-down report of each vehicle type with its details on repair count, parts costs, labor costs, total costs and details.
- Case “Below Cost Sales”
    - Read all *Vehicle* entities
    - Filter out the entities whose *SoldPrice* < *InvoicePrice*
    - For each, read the salesperson’s name and customer’s name (first/last for individuals or business name for businesses) from *User* and *Customer* entities with key *SalespeopleUsername* and *CustomerId* respectively.
    - Query to retrieve the record

```
SELECT DateSold, InvoicePrice, SoldPrice,
ROUND(SoldPrice/InvoicePrice * 100) AS ProfitRatio,
COALESCE(CONCAT(I.FirstName, ' ',I.LastName), BusinessName) AS
CustomerName, CONCAT(U.FirstName, ' ',U.LastName) AS SalespersonName
FROM Vehicle
LEFT JOIN Individual AS I ON Vehicle.CustomerID = I.CustomerID
LEFT JOIN Business AS B ON Vehicle.CustomerID = B.CustomerID
LEFT JOIN User AS U ON Vehicle.SalespeopleUsername = U.Username
WHERE DateSold IS NOT NULL AND SoldPrice < InvoicePrice
ORDER BY DateSold DESC, ProfitRatio DESC;
```

    - CustomerName will automatically contain either the BusinessName or the Individual’s FirstName and Last depending on the Customer type.
    - For a sale where *SoldPrice* to *InvoicePrice* ratio is less than or equal to 95%, the background of that row should be highlighted red. Display the report by sales date descending and ratio descending.
- Case “Average Time in Inventory”
    - Query each of the vehicle subclass table joined with the main vehicle table to retrieve sales information .
    - Using the query to compute the average amount of time sold vehicles remain in inventory (*DateSold + 1 - DateAdded*) using the DATEDIFF function.
    - Queries for each subclass table.

```
SELECT 'Car' as VehicleType, AVG(DATEDIFF(DateSold+1, DateAdded))
FROM Vehicle_Car
JOIN Vehicle ON Vehicle_Car.VIN = Vehicle.VIN
WHERE Vehicle.CustomerID IS NOT NULL
UNION
SELECT 'Truck' as VehicleType, AVG(DATEDIFF(DateSold+1, DateAdded))
FROM Vehicle_Truck
JOIN Vehicle ON Vehicle_Truck.VIN = Vehicle.VIN
WHERE Vehicle.CustomerID IS NOT NULL
UNION
SELECT 'SUV' as VehicleType, AVG(DATEDIFF(DateSold+1, DateAdded))
```

```
FROM Vehicle_SUV
JOIN Vehicle ON Vehicle_SUV.VIN = Vehicle.VIN
WHERE Vehicle.CustomerID IS NOT NULL
UNION
SELECT 'Convertible' as VehicleType, AVG(DATEDIFF(DateSold+1,
DateAdded))
FROM Vehicle_Convertible
JOIN Vehicle ON Vehicle_Convertible.VIN = Vehicle.VIN
WHERE Vehicle.CustomerID IS NOT NULL
UNION
SELECT 'Van' as VehicleType, AVG(DATEDIFF(DateSold+1, DateAdded))
FROM Vehicle_Van
JOIN Vehicle ON Vehicle_Van.VIN = Vehicle.VIN
WHERE Vehicle.CustomerID IS NOT NULL;
```

    - Display the report (if a vehicle type has no sales history, the query will return empty and the report should display “N/A” for that vehicle type).
- Case “Part Statistics”
    - Read all *Parts* entities
    - Grouped by *VendorName*, sum the *QuantityUsed* and *UnitPrice times QuantityUsed*.
    - Query

```
SELECT VendorName, SUM(QuantityUsed) AS NumberOfParts,
SUM(QuantityUsed * UnitPrice) AS TotalSpent
FROM Part
GROUP BY VendorName;
```

    - Using those stats, display each vendor: the vendor’s name, the number of parts supplied by that vendor, and the total dollar amount spent on parts.
- Case “Monthly Sales”
    - Read all *Vehicle* entities
    - Filter out the sold entities and sorted by *DateSold* descending
    - Using a window side of a year and a month respectively, slice the whole list into a list of windows (each contains a list of sold vehicles).
    - For either yearly and monthly window, drop if empty, count the number, sum the *SoldPrice* for total sales income, the *InvoicePrice* for total invoice price. The net income is total sales income less total invoice price. Keep a “total sales income/total invoice price” ratio.
      If the ratio is greater than or equal to 125%, its row should be highlighted with a green background. If the ratio is less than or equal to 110%, it should be highlighted with a yellow background.
    - Query to fetch by year

```
SELECT YEAR(DateSold) as Year, COUNT(VIN) AS TotalVehiclesSold,
SUM(SoldPrice) AS TotalSalesIncome, SUM(SoldPrice) -
SUM(InvoicePrice) AS TotalNetIncome, ROUND(SUM(SoldPrice) /
SUM(InvoicePrice) * 100) AS ProfitRatio
FROM Vehicle
WHERE CustomerID IS NOT NULL
GROUP BY YEAR(DateSold)
ORDER BY YEAR(DateSold) DESC;
```

- Query to fetch by year and month

```
SELECT YEAR(DateSold) as Year, MONTH(DateSold) AS Month, COUNT(VIN)
AS TotalVehiclesSold, SUM(SoldPrice) AS TotalSalesIncome,
SUM(SoldPrice) - SUM(InvoicePrice) AS TotalNetIncome,
ROUND(SUM(SoldPrice) / SUM(InvoicePrice) * 100) AS ProfitRatio
FROM Vehicle
WHERE CustomerID IS NOT NULL
GROUP BY YEAR(DateSold), MONTH(DateSold)
ORDER BY YEAR(DateSold) DESC, MONTH(DateSold) DESC;
```

- Display the report with the total number of vehicles sold, the total sales income, the total net income, and ratio as a percentage (such as 125%), for each yearly and monthly. (first part)
- Using the yearly and monthly window sliced results in the earlier step, given any year or month window, Grouped by *SalespeopleUsername,* count the number of records in each person and sum the *SoldPrice.* Read the salesperson's first and last name from *User* with key *SalespeopleUsername.*
- Display the drilldown report with the salesperson's first and last name, the number of vehicles they sold in that year and month and their total sales for that year and month, sorted by total vehicles descending followed by total sales descending. (second part)
- Query to fetch drill down by selected year

```
SELECT FirstName, LastName, COUNT(VIN) AS TotalVehiclesSold,
SUM(SoldPrice) AS TotalSales
FROM Vehicle JOIN User ON Vehicle.SalespeopleUsername =
User.Username WHERE YEAR(DateSold) = '$Year'
GROUP BY Username
ORDER BY TotalVehiclesSold DESC, TotalSales DESC
LIMIT 1;
```

- Query to fetch drill down by selected year and month

```
SELECT FirstName, LastName, COUNT(VIN) AS TotalVehiclesSold,
SUM(SoldPrice) AS TotalSales
```

```
FROM Vehicle JOIN User ON Vehicle.SalespeopleUsername =
User.Username
WHERE MONTH(DateSold) = '$Month' AND YEAR(DateSold) = '$Year'
GROUP BY Username
ORDER BY TotalVehiclesSold DESC, TotalSales DESC
LIMIT 1;
```