[**EDIT LINK**](https://docs.google.com/document/d/1RpSXNKfcEtZoY0WcfEuECQYgKTbJryUMMtfPkiq81gQ/edit)

# **Table of Contents:**

[Search page/Display Main Menu](#_gx9a2qpqtgmn)

[View Vehicle Details](#_atenskhoqk2g)

[Login](#_rboizwxt7ve4)

[Sales order](#_52zrbs2yaehl)

[Repair](#_pxoziee9h5iz)

[Sales by Color](#_x9xztswvlfne)

[Sales by Type](#_rct6iao21r0b)

[Sales by Manufacturer](#_pq87cxqr076g)

[Gross Customer Income](#_qx21rqk9kref)

[View Drill-Down Customers](#_t08i4qkllude)

[Repairs by Manufacturer/Type/Model](#_b9pu53hrgggl)

[View Drill-Down Manufacturer](#_to6176i60ykh)

[Below Cost Sales](#_fzq3eb21n2qk)

[Average Time in Inventory](#_pyg9qi5upwo9)

[Parts Statistics](#_t4cnlnjuphvl)

[Monthly Sales](#_4x5vq092zfuq)

[Monthly Sales Drill-Down](#_5eg3ocw6vukg)

**Screen/Page**

**entity**

**input field**

***Button***

‘$session variable’

**Relationship**

DARK Blue: Database table names in lowercase

Grey shadow part is handled by system instead of sql

# **Searching for vehicles/Display Main Menu**

Abstract Code

* (Initial state of the application and upon ***Return to Main Menu*** button clicks.)
* Clear all input fields
* Disable additional features unless the user is already login(‘$UserType’ is not null or anonymous).
* Show ***search*** and ***login*** button on the **Main Menu screen**
* Query for information of counts of Vehicle using Vehicle.VIN, show this information as the total number of vehicles available for purchase in the system in the **Main Menu screen**

|  |
| --- |
| SELECT COUNT(Vehicle) AS 'Number of Vehicles'  FROM Vehicle; |

* Find available choices of searching input fields ( Manufacturer, Model year, Model, Color) (available Vehicle Type is handle by system) and fill available choices in the dropdowns for input fields in the **Main Menu screen**

|  |
| --- |
| SELECT DISTINCT (year)  FROM Vehicle;  SELECT DISTINCT (model)  FROM Vehicle;  SELECT DISTINCT (color)  FROM VehicleColor;  SELECT DISTINCT mf\_name  FROM Manufacturer; |

* All users are able to fill in **List price** and **Keyword** input fields. Users also are able to select a **vehicle type**/**Manufacturer**/**Model**/**Year**/**Color/VIN(if enabled)** from the dropdown list in each input field in the **Main Menu screen**
* Upon:
  + Click the ***search*** button:
    - Read **Type**, **manufacturer**, **model**, **year**, **color,** **list price**, **keyword, VIN(if enabled)** input fields from the **Main Menu screen**
    - if data is valid and if there are **Vehicle** that matches the search criteria:
      * Display the matching **Vehicle** as a list sorted by **Vehicle** VIN in ascending order on **Main Menu screen**

|  |
| --- |
| SELECT vin, type, model,year, mf\_name, description  FROM Vehicle  WHERE vin='$Vin'  AND type='$Type'  AND (model='$Model' or model LIKE '%$KeyWord%')  AND (year='$Year' or year LIKE '%$KeyWord%')  AND (mf\_name='$MfName' or mf\_name LIKE '%$KeyWord%')  AND description LIKE '%KeyWord%'  ORDER BY vin ASC;  for each result $vin  SELECT Color  FROM VehicleColor  Where vin='$vin'  end for |

* + - * Session variable ‘$vehiclevin’ = Vehicle VIN
      * Session variable ‘$vehicleType’ = Vehicle Type
      * Users are allowed to select an individual result from the list. If the user selects a **Vehicle**:
        + Jump to the **View Vehicle Details** task.
    - Otherwise, display the message “Sorry, it looks like we don’t have that in stock!” on the **Main Menu screen**
  + Click the ***login*** button: Jump to the **Login** task:
    - If ‘$UserType’ returned:
      * ‘$UserType’==’Inventory clerks’, enable ***Add Vehicle*** button and **new vehicle form** on **Main Menu screen**
      * ‘$UserType’==’Salespeople’, enable input field **VIN** in searching criteria on **Main Menu screen**
      * ‘$UserType’==’Service Writer’, enable ***repair form*** button on **Main Menu screen**
      * ‘$UserType’==’Managers’, enable ***report*** button, input field **VIN** in searching criteria, option to filter by soldvehicles, unsold vehicles, or all vehicles, Dropdown on **Main Menu screen**
      * ‘$UserType’==’Roland Around’, enable all the features on **Main Menu screen**
  + Click the ***Add Vehicle*** button:
    - User will fill the **VIN**, **vehicle type**, **invoice price**, etc., along with the **date** it was added to inventory in **new vehicle form** on **Main Menu screen**.
    - Read those values from input fields and if Data is valid and **VIN** does not already exist as a Vehicle.VIN:
      * Inert new **Vehicle** instance with those values, then clear any success/error message, display a success message, ‘$vehiclevin’ = **VIN** and call the **View Vehicle Details** task.

|  |
| --- |
| INSERT INTO Vehicle(vin, type, model, year, mf\_name, decription)  VALUES ('$Vin', '$Type', '$Model', '$Year', '$MfName', '$Description');  INSERT INTO VehicleColor(vin, color)  VALUES ('$Vin', '$Color');  if '$VehicleType'=='SUV'  INSERT INTO SUV(vin, number\_of\_cupholders, drive\_train\_type)  VALUES ('$Vin', '$NumberOfCupholders', '$DriveTrainType');  elif '$VehicleType'=='Van'  INSERT INTO Van(vin, has\_driversidebackdoor)  VALUES ('$Vin', '$HasDriverSideBackDoor');  elif '$VehicleType'=='Truck'  INSERT INTO Truck(vin, cargo\_capacity, cargo\_covertype, no\_rearaxles)  VALUES ('$Vin', '$CargoCapacity', '$CargoCovertype', '$NoRearaxles');  elif '$VehicleType'=='Convertible'  INSERT INTO Convertible(vin, back\_seat\_count, roof\_type)  VALUES ('$Vin', '$BackSeatCount', '$RoofType');  else '$VehicleType'=='Car'  INSERT INTO Car(vin, number\_of\_doors)  VALUES ('$Vin', '$NumberOfDoors'); |

* + Click the ***repair form*** button: Jump to the **Repair** task
  + Click the ***report*** button: Read the choice from the report dropdown menu on **Main Menu screen** then call the corresponding report task

# **View Vehicle Details**

Abstract Code

* User selected on Vehicle(‘$VehicleVin’ and it’s ‘$vehicleType’) from the list on the **Main Menu screen**
* Display **Detail Page** Screen
* Enable link to ***sell the vehicle*** on **Detail Page** Screen if current user is SalesPerson
* Query for information about the **Vehicle** and it’s details using ‘$vehiclevin’ from the HTTP Session/Cookie.:
  + Display ‘$vehiclevin’ on the **Detail Page**
  + Find and display the **Vehicle**.Type on the **Detail Page**
  + Find and display the attributes of the **Vehicle**.Type on the **Detail Page**
  + Find and display the **Vehicle**.Model and Vehicle.Year on the **Detail Page**
  + Find and display the **Vehicle**.Manufacturer on the **Detail Page**
  + Find and display the **Vehicle**.Color on the **Detail Page**
  + Find **Vehicle**.InvoicePrice and display invoice price times 125% as list price on the **Detail Page**
  + Find and display the **Vehicle**.VDescription on the **Detail Page**
  + If ‘$UserType’==‘Inventory clerks’:
    - Find and display the **Vehicle**.invoice\_price on the **Detail Page**

|  |
| --- |
| SELECT v.vin, v.description, v.type, v.year, v.model, v.mn\_name, v.invoice\_price, v.invoice\_price\*1.25 as list\_price, t.’$attributes’  FROM Vehicle v INNER JOIN '$VehicleType' t on v.Vin=t.Vin  WHERE v.Vin='$VehicleVin';  SELECT Color  FROM VehicleColor  Where vin='$VehicleVin' |

* Upon:
  + Click ***sell the vehicle*** link: Call the **sales order**.
  + Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

###### **Login**

Abstract Code

* User click ***login*** button on the **Main Menu screen**
* Display **Login form** Screen
* User enters ‘$UserName’, ‘$password’ input fields.
* If data validation is successful for both username and password, then:
  + When ***Enter*** button is clicked:
    - If User record is found but password not match **PrivilegedUser** password:
      * Go back to **Login form**, with error message.

|  |
| --- |
| for each $userType ['ServiceWriter','Manager,Owner','SalesPerson','InventoryClerk']  SELECT p.password  FROM '$userType' t INNER JOIN PrivilegedUser p  ON p.username=t.username  WHERE p.username='$Username'  end for |

* + - else:
      * Store login information as session variable ‘$UserType’
      * Call the Display Main Menu task with ‘$UserType’.
    - Else username and password input fields are invalid, display Login form, with error message.

### 

### **Sales order**

Abstract Code:

* user click ***sell the vehicle*** on **Detail Page**
* **Sales order Form** is Displayed.
* user will fill the customer profile inputs field showed on **Sales order Form**:
  + if customer is an individual: fill their **first** and **last names**, along with their **driver’s license number**
  + if customer is a business: fill the business’ **tax identification number** and **business name**, along with the **name of a primary contact** and their **title**
* user will fill the transaction detail field including: **Vehicle’s VIN**, **sold price**, **sold date**
* upon:
  + click the ***lookup*** button on **Sales order Form**: run the **lookup customer** task by query customer with **driver’s license number or tax identification number**
    - if a customer is not found:
      * Read customer profile input fields, and call **Add customer** task.

|  |
| --- |
| if '$customer'=='indivial'  INSERT IGNORE INTO Individual  SET driver\_license\_number = '$DriverLicenseNumber',  first\_name = '$FirstName',  last\_name = '$LastName',  address = '$Address',  phone\_number='$PhoneNumber',  email\_address='$EmailAddress';  if '$customer'=='business'  INSERT IGNORE INTO Business  SET tin = '$Tin',  bname = '$BName',  pcname = '$PCName',  title = '$Title',  address = '$Address',  phone\_number='$PhoneNumber',  email\_address='$EmailAddress'; |

* + click the confirm sale button on **Sales order Form**: run the **confirm sale** task by query vehicle’s invoice price by reading and using **Vehicle’s VIN**
    - if **sold price** is less than or equal to 95% of **Vehicle**.invoice\_price:
      * Display error massage of rejecting sale.

|  |
| --- |
| SELECT invoice\_price\*0.95 as minimum\_price  FROM Vehicle  WHERE Vin='$VehicleVin'; |

* + - Otherwise, insert new **sale** instance with those values including **sold price**, **sold date**,customer **driver’s license number** or **tax identification number**, **Vehicle’s VIN**, **SalesPerson’s Name.** Display a success message.

|  |
| --- |
| if '$customer'=='individual'  INSERT INTO VehicleSoldIndividual(vin,driver\_license\_number,salesPerson\_username,sale\_date,sold\_price)  VALUES ('$Vin','$DriverLicenseNumber','$SalesPersonUsername','$SaleDate','SoldPrice');  if '$customer'=='business'  INSERT INTO VehicleSoldBusiness(vin,tin,salesPerson\_username,sale\_date,sold\_price)  VALUES ('$Vin','$Tin','$SalesPersonUsername','$SaleDate','SoldPrice'); |

* + Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

###### **Repair**

Abstract Code

* User click ***repair form***  button on the **Main Menu screen**
* Partial **repair form** is displayed.
* User will fill the **VIN** input field.
* If data validation is successful for **VIN**, then:
  + When ***Enter*** button is clicked: run **search vin** task.
    - If the **VIN** does not match a **Vehicle**.VIN in the database:
      * Display an error message
    - Otherwise, the rest of the **repair form** will be displayed:
      * Run **View Vehicle Details** task, the results will be displayed on **repair form screen**

|  |
| --- |
| SELECT v.vin, v.type, v.year, v.model, v.mn\_name  FROM Vehicle v INNER JOIN (SELECT \* FROM VehicleSoldIndividual UNION ALL SELECT \* FROM VehicleSoldBusiness) sold  ON v.vin=sold.vin  WHERE v.vin='$VehicleVin';  SELECT Color  FROM VehicleColor  Where vin='$VehicleVin'; |

* + - * Check if the **Vehicle** is associated with a **repair** order.
      * If vin is not null and completion\_date is null then enable only **update button**.

|  |
| --- |
| SELECT vin, completion\_date  FROM Repair  WHERE vin='$Vin'; |

* + - * + If no repairs are open for the **Vehicle**:

display **add repair button**

run **add repair** task after user click ***add repair***

button

Insert new **repair** order instance with **odometer reading** input field filled by user

click the ***lookup*** button on **Sales order Form**: run the **lookup customer** task by query customer with **driver’s license number or tax identification number**

|  |
| --- |
| if '$customer'=='individual'  SELECT driver\_license\_number  FROM IndividualNeedsRepair  WHERE driver\_license\_number='$DriverLicenseNumber';  if '$customer'=='business'  SELECT TIN  FROM BusinessNeedsRepair  WHERE tin='$tin'; |

if a customer is not found:

Read customer profile input fields, and call **Add customer** task. add the customer to the system for repair order

|  |
| --- |
| if '$customer'=='individual'  INSERT INTO IndividualNeedsRepair(vin,driver\_license\_number)  VALUES ('$Vin','$DriverLicenseNumber');  if '$customer'=='business'  INSERT INTO BusinessNeedsRepair(vin,tin)  VALUES ('$Vin','$Tin'); |

User are allowed to fill the inputs field of **labor charge** and **parts**:

input field **quantity**, **vendor**, **part number**, **price** can be filled by user.

If Data is valid and user click ***add parts*** button, insert new parts instance with those values with the service writer’s name, display a success message, Otherwise, display an appropriate error message. run **add part task to** add the part to current **repair** order

if ***add labor charge*** button is clicked, add the labor charge to current **repair** order

|  |
| --- |
| INSERT INTO Repair(vin, start\_date, description, odometer\_reading, labor\_charges)  VALUES ('$Vin','$StartDate','$Description','$OdometerReading','$LaborCharges');  INSERT INTO Part(vin,part\_number,vendor\_name,quantity,price)  VALUES ('$Vin','$PartNumber','$VendorName','$Quantity','$Price'); |

* + - * + Otherwise, upon:

click ***updating labor charges*** button: read input field labor charges and run **update repair** task

|  |
| --- |
| UPDATE Repair  SET labor\_charges='$LaborCharges'  WHERE vin='$Vin'; |

click ***adding parts*** button: read input field for parts and run **add part** task

|  |
| --- |
| INSERT IGNORE INTO Part(vin,part\_number,vendor\_name,quantity,price)  VALUES ('$Vin','$PartNumber','$VendorName','$Quantity','$Price'); |

click ***completing*** button: add **repair**.completion\_date to current **repair** as current date

|  |
| --- |
| UPDATE Repair  SET completion\_date='$CurrentDate'  WHERE vin='$Vin'; |

* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### **Sales by Color**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about each **VehicleSold**, find the **VehicleSold** SoldDate and corrsponding sold **Vehicle** color using **Vehicle** VIN. **VehicleSold** is the **sold relationship** among **Vehicle, Customer,** and **SalesPerson**. Count the number of **VehicleSold** based on the different periods (**VehicleSold** SoldDate) and group the count by different colors. Put count and color data in a table:
  + Each color is one row.
  + Columns are the count of sales.
  + Columns including sales in the previous 30 days, sales in the previous year, sales overall time.
  + If a color does not have any sales, it is shown with a value of “0”.

|  |
| --- |
| SELECT one.color,one.count as previous\_30days,two.count as previous\_year,three.count as overtime  FROM (((SELECT c.color, count(sold.vin)  FROM VehicleColor c LEFT JOIN (SELECT vin FROM VehicleSoldIndividual UNION ALL SELECT vin FROM VehicleSoldBusiness) sold  ON sold.vin=c.vin  WHERE DATEDIFF(day, CURRENT\_TIMESTAMP, sold.sale\_date)<=30  GROUP BY c.color) one  INNER JOIN (SELECT c.color, count(sold.vin)  FROM VehicleColor c LEFT JOIN (SELECT vin FROM VehicleSoldIndividual UNION ALL SELECT vin FROM VehicleSoldBusiness) sold  ON sold.vin=c.vin  WHERE DATEDIFF(day, CURRENT\_TIMESTAMP, sold.sale\_date)<=365  GROUP BY c.color) two ON one.color = two.color)  INNER JOIN (SELECT c.color, count(sold.vin)  FROM VehicleColor c LEFT JOIN (SELECT vin FROM VehicleSoldIndividual UNION ALL SELECT vin FROM VehicleSoldBusiness) sold  ON sold.vin=c.vin  GROUP BY c.color) three ON one.color = three.color)  ORDER BY one.color ASC; |

* Display the table in the **Sales by Color report**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### 

### **Sales by Type**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about each **Sale**, find the **Sale**.PurchaseDate and corresponding sold **Vehicle**.type using **Vehicle**.VIN. **Sale** and **Vehicle** are related by **transfers ownership of** relationship. Count the number of **Sale** based on the different periods (**Sale**.PurchaseDate) and group the count by different types. Put count and type data in a table:
  + Each type is one row.
  + Columns are the count of sales.
  + Columns including sales in the previous 30 days, sales in the previous year, sales overall time.
  + If a type does not have any sales, it is shown with a value of “0”.

|  |
| --- |
| SELECT one.type,one.count as previous\_30days,two.count as previous\_year,three.count as overtime  FROM (((SELECT t.type, count(sold.vin)  FROM Vehicle t LEFT JOIN (SELECT vin FROM VehicleSoldIndividual UNION ALL SELECT vin FROM VehicleSoldBusiness) sold  ON sold.vin=t.vin  WHERE DATEDIFF(day, CURRENT\_TIMESTAMP, sold.sale\_date)<=30  GROUP BY t.type) one  INNER JOIN (SELECT t.type, count(sold.vin)  FROM Vehicle t LEFT JOIN (SELECT vin FROM VehicleSoldIndividual UNION ALL SELECT vin FROM VehicleSoldBusiness) sold  ON sold.vin=t.vin  WHEREDATEDIFF(day, CURRENT\_TIMESTAMP, sold.sale\_date)<=365  GROUP BY t.type) two ON one.type = two.type)  INNER JOIN (SELECT t.type, count(sold.vin)  FROM Vehicle t LEFT JOIN (SELECT vin FROM VehicleSoldIndividual UNION ALL SELECT vin FROM VehicleSoldBusiness) sold  ON sold.vin=t.vin  GROUP BY t.type) three ON one.type = three.type)  ORDER BY one.type ASC; |

* Display the table in the **Sales by Type report**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### 

### **Sales by Manufacturer**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about each **Sale**, find the **Sale**.PurchaseDate and corresponding sold **Vehicle**.Manufacturer using **Vehicle**.VIN. **Sale** and **Vehicle** are related by **transfers ownership of** relationship. Count the number of **Sale** based on the different periods (**Sale**.PurchaseDate) and group the count by different Manufacturers. Put count and Manufacturer data in a table:
  + Each Manufacturer is one row.
  + Columns are the count of sales.
  + Columns including sales in the previous 30 days, sales in the previous year, sales overall time.
  + If a type does not have any sales, it will not be put on the table.

|  |
| --- |
| SELECT one.mf\_name,one.count as previous\_30days,two.count as previous\_year,three.count as overtime  FROM (((SELECT m.mf\_name, count(sold.vin)  FROM Vehicle m LEFT JOIN (SELECT vin FROM VehicleSoldIndividual UNION ALL SELECT vin FROM VehicleSoldBusiness) sold  ON sold.vin=m.vin  WHERE DATEDIFF(day, CURRENT\_TIMESTAMP, sold.sale\_date)<=30  GROUP BY m.mf\_name) one  INNER JOIN (SELECT m.mf\_name, count(sold.vin)  FROM Vehicle m LEFT JOIN (SELECT vin FROM VehicleSoldIndividual UNION ALL SELECT vin FROM VehicleSoldBusiness) sold  ON sold.vin=m.vin  WHERE DATEDIFF(day, CURRENT\_TIMESTAMP, sold.sale\_date)<=365  GROUP BY m.mf\_name) two ON one.mf\_name = two.mf\_name)  INNER JOIN (SELECT m.mf\_name, count(sold.vin)  FROM Vehicle m LEFT JOIN (SELECT vin FROM VehicleSoldIndividual UNION ALL SELECT vin FROM VehicleSoldBusiness) sold  ON sold.vin=m.vin  GROUP BY m.mf\_name) three ON one.mf\_name = three.mf\_name)  ORDER BY one.mf\_name ASC; |

* Display the table in the **Sales by Type report**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### 

### 

### 

### 

### 

### 

### 

### **Gross Customer Income**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about all **Sale** and **Repair**, group **Sale** and **Repair** the by **Customer**’s driver’s license number or tax identification number. And sum **Sale**.sold\_price and **Repair** total cost as gross income for each **Customer** ID. Both **Sale** and **Repair** have relationship with **Customer**. Sort Customer ID by gross income and keep the largest 15 one. Find and Place all following data for each one of 15 Customer in a list:
  + - **Customer**’s name
    - The date of the first sale or repair start date
    - the date of the most recent sale or repair start date
    - The number of sales
    - The number of repairs
    - the Gross income
  + The list of **customer**s will be by gross income descending and by last sale/repair start date descending.

|  |
| --- |
| SELECT f.number\_sale,f.number\_repair,f.first\_start\_date,f.last\_start\_date,f.first\_sale\_date,f.last\_sale\_date,f.id,f.gross\_income,  i.first\_name,i.last\_name, b.bname  FROM (select count(s.vin) number\_sale, count(r.vin) number\_repair, min(r.start\_date) first\_start\_date,max(r.start\_date) last\_start\_date,min(s.sale\_date) first\_sale\_date,max(s.sale\_date) last\_sale\_date,s.id, (sum(s.sold\_price)+sum(r.total\_repair\_cost)) gross\_income  from (select vin, driver\_license\_number as id,sold\_price,sale\_date from VehicleSoldIndividual union all select vin, tin as id, sale\_date,sold\_price from VehicleSoldBusiness)s left join (select r.vin, r.start\_date,(r.labro\_charges+temp.total\_part\_cost) as total\_repair\_cost from Repair r inner join (select vin,sum(quantity\*price) as total\_part\_cost from part group by vin)temp on r.vin=temp.vin)r on s.vin=r.vin  GROUP BY s.id  ORDER BY income\_per\_car DESC  LIMIT 15) f  LEFT JOIN Individual i  ON f.id = i.driver\_license\_number  LEFT JOIN Business b  ON f.id = b.tin  ORDER BY f.gross\_income DESC,f.last\_sale\_date DESC,f.last\_start\_date DESC; |

* + Display the list in the **Gross Customer Income report**
  + Users are able to select one ***customer’s name*** in the list. If User clicks one of the ***customer’s name***: Jump to **View Drill-Down** task with selected **Customer.** and their **sale** and **repair** which can be get from the table that group **sale** and **repair** by customer.
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

# 

# 

# 

# **View Drill-Down Customers**

Abstract Code

* User selected a customers from the list on the **Gross Customer Income report**
* Retrieve the Customer ID, Sales, repairs from the **Gross Customer Income** task.
* Find and place all following data in a list, each row is for one **Sale**:
  + **Sale**.SoldDate
  + **Sale**.SoldPrice
  + Find the **Vehicle** associated with **Sale**, and get the **Vehicle**.VIN
  + **Vehicle**.manufactuer
  + **Vehicle**.model
  + **Sale**.salespersonname
* The listing should be sorted by sale date descending and VIN ascending

|  |
| --- |
| SELECT v.vin, v.year, v.model, v.mn\_name, sold.sale\_date, sold.sales\_person\_username, sold\_price  FROM Vehicle v INNER JOIN (select vin, driver\_license\_number as id,sold\_price,sale\_date,sales\_person\_username from VehicleSoldIndividual union all select vin, tin as id, sale\_date,sold\_price,sales\_person\_username from VehicleSoldBusiness) sold  ON v.vin=sold.vin  WHERE sold.id='$Customer\_id'  ORDER BY sold.sale\_date DESC, sold.vin ASC; |

* Display the list in the **section for vehicle sales** on the **Drill-Down Screen**
* Find and place all following data in a list, each row is for one **Repair**:
  + **repair**.start\_date
  + **repair**.complete\_date if available
  + Find the **Vehicle** associated with **Repair**, and get the **Vehicle**.VIN
  + **repair.**odometer
  + **repair**.labor cost
  + parts cost
  + total cost
  + the service writer who opened the repair
* This listing should be 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.

|  |
| --- |
| SELECT \*  FROM (  SELECT startdate, completiondate, **Repair.**vin, odometerreading, laborcharges, SUM(quantity\*price), SUM(quantity\*price+laborcharges)  FROM **Repair** INNER JOIN **Part**  WHERE **Repair**.vin=**Part**.vin  ORDER BY startdate DESC, completiondate DESC NULLS FIRST, **Repair**.vin ASC;  ) t  order by  CASE  WHEN completiondate IS NULL THEN 1  ELSE 2  END; |

* Display the list in the **section for repairs** on the **Drill-Down Screen**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### **Repairs by Manufacturer/Type/Model**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about each **Repair**, Count the number of **Repair**, the sum of all parts cost, the sum of all labor cost, and the sum of total repair costs, including any repairs in progress for each **Vehicle**.Manufacturer. The **Vehicle**.Manufacturer is found by using **Vehicle**.VIN associate with each **repair**.
* Populate these data in a list where each row is for one **Vehicle**.Manufacturer, Manufacturers whose vehicles do not have any repairs should be listed on this list, and the list should be sorted by manufacturer name ascending.

|  |
| --- |
| SELECT COUNT(**Repair**), SUM(quantity\*price) AS partscost, SUM(laborcharges) AS laborcost, partscost UNION laborcost AS repaircosts, mfname  FROM **Repair** INNER JOIN **Part**  WHERE **Repair**.vin=**Part.**vin  ORDER BY mfname ACS; |

* Display the list on **Repairs by Manufacturer/Type/Model screen**
* Users are able to select one manufacturer’s name from the list
  + Run **Drill-down** task with the manufacturer’s name

# 

# 

# 

# 

# 

# **View Drill-Down Manufacturer**

Abstract Code

* User select a manufacturer from the list on the **Repairs by Manufacturer/Type/Model screen**
* Retrieve the manufacturer, **Vehicle**, **repairs** from the **Repairs by Manufacturer/Type/Model** task.
* Find and place all following data in a list, each row is for one **Vehicle**.Type:
  + repair count
  + parts costs
  + labor costs
  + total costs

|  |
| --- |
| SELECT COUNT(**Repair**), SUM(quantity\*price) AS partscost, partscost+laborcharges AS totalcosts  FROM **Vehicle** JOIN **Repair** JOIN Part  WHERE **Vehicle**.vin=**Repair**.vin  AND **Repair**.vin=**Part**.vin  GROUP BY **Vehicle**.type; |

* Find and place all following data in a list, each row is for one **Vehicle**.model:
  + repair count
  + parts costs
  + labor costs
  + total costs
* Lists are sorted by repair count descending(by vehicle type sorted first, and then detail rows sorted).

|  |
| --- |
| SELECT COUNT(**Repair**), SUM(quantity\*price) AS partscost, partscost+laborcharges AS totalcosts  FROM **Vehicle** JOIN **Repair** JOIN Part  WHERE **Vehicle**.vin=**Repair**.vin  AND **Repair**.vin=**Part**.vin  GROUP BY **Vehicle**.type; |

* Display the list on the **Drill-Down Screen**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### 

### **Below Cost Sales**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about each **Sale**, find the **Sale** that **Sale**.invoice\_price > **Sale**.sold\_price. Find and place all following data in a list, each row is for one **Sale**:
  + **Sale**.completedate
  + **Sale**.invoice price
  + **Sale**.sold price
  + sold price/invoice price ratio as a percentage
  + **customer**.name. Retrieve customer’s name by **customer** ID associate with **Sale**.
  + **Sale**.salesperson’s firstname and lastname
* For a sale whose ratio is less than or equal to 95%, the background of that row should be highlighted red. Sales should be listed by sales date descending and ratio descending.

|  |
| --- |
| SELECT saledate, invoiceprice, soldprice, soldprice/invoiceprice AS salesratio,firstname, lastname, ufirstname, ulastname  FROM **VehicleSoldIndividual** INNER JOIN **Vehicle** INNER JOIN **Individual** INNER JOIN **InventoryClerk** INNER JOIN **PrevilegedUser**  WHERE invoiceprice>soldprice  AND **VehicleSoldIndividual**.vin=**Vehicle**.vin  AND **VehicleSoldIndividual**.driverlicensenumber=**Individual**.driverlicensenumber  AND **VehicleSoldIndividua**l.salespersonusername=**Salesperson**.username  AND **Salesperson**.username=**PrevilegedUser**.username  ORDER BY saledate DESC, salesratio DESC;  UNION  SELECT saledate, invoiceprice, soldprice, soldprice/invoiceprice AS salesratio,firstname, lastname, ufirstname, ulastname  FROM **VehicleSoldBusiness** INNER JOIN **Vehicle** INNER JOIN **Business** INNER JOIN **InventoryClerk** INNER JOIN **PrevilegedUser**  WHERE invoiceprice>soldprice  AND **VehicleSoldBusiness**.vin=**Vehicle**.vin  AND **VehicleSoldBusiness**.tin=**Business**.tin  AND **VehicleSoldBusiness**l.salespersonusername=**Salesperson**.username  AND **Salesperson**.username=**PrevilegedUser**.username  ORDER BY saledate DESC, salesratio DESC; |

* Display the table in the **Below Cost Sales report**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### 

### 

### 

### 

### 

### 

### **Average Time in Inventory**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about each **Sale**, find the **Sale**.complete date. Find the Vehicle.Date using VehicleVIN associated with Sale. Calculate the difference between **Sale**.complete date and **Vehicle**.Date as the amount of time a vehicle remains in inventory group by Vehicle.Type. Calculate and Put the average amount of time a vehicle remains in inventory in a list, each row is for one Vehicle.Type
* If a **vehicle**.type has no sales history, the report should display “N/A” for that **vehicle**.type.

|  |
| --- |
| SELECT saledate, dateadded, DATEDIFF(day, saledate, dateadded) AS vehicledate, COUNT(Vehicle.vin) AS vehiclenums, vehicledate/vehiclenums AS avgtime  FROM **VehicleSoldIndividual** INNER JOIN **Vehicle**  WHERE **VehicleSoldIndividual**.vin=**Vehicle**.vin  GROUP BY **Vehicle**.type; |

* Display the table in the **Average Time in Inventory report**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### 

### 

### 

### 

### 

### 

### 

### 

### **Parts Statistics**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about each **Part**, find the **Part**.price and **Part**.quantity group by **Part**.vendorname. Calculate the total cost and total quantity of part for each **Part**.vendorname. Then put the vendor’s name, the number of parts supplied by that vendor, and the total dollar amount spent on parts in a list. The list should be sorted by total dollar amount spent descending.

|  |
| --- |
| SELECT vendorname, SUM(quantity) AS totalquantity, SUM(price\*quantity) AS totalcost  FROM **Part**  GROUP BY vendorname  ORDER BY totalcost DESC; |

* Display the list in the **Parts Statistics report**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **Monthly Sales**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about **Sale**.
  + Group the Sale by year and month based on **Sale**.Date then calculate the count and sum for each group.

|  |
| --- |
| SELECT COUNT(**VehicleSoldIndividual**) ALL UNION COUNT(**VehicleSoldBusiness**) ALL  FROM **VehicleSoldIndividual** INNER JOIN **VehicleSoldBusiness**  GROUP BY year(saledate), month(saledate); |

* + create a list which has:
    - the total number of vehicles sold, the total sales income, the total net income (calculate by using soldprice - invoice price), and the sold price/invoice price ratio as a percentage (such as 125%) for each year and month based on **Sale.**Date.
  + If a year or month does not have sales data, it can be excluded from this report.
  + When the ratio for a month 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.
  + The results will be ordered by year and month descending, with the most recent year and month as the first result.

|  |
| --- |
| SELECT COUNT(**VehicleSoldIndividual**) AS individualsale, COUNT(**VehicleSoldBusiness**) AS businesssale, individualsale+businesssale AS totalsale, SUM(**VehicleSoldIndividual**.soldprice) AS indincome, SUM(**VehicleSoldBusiness**.soldprice) AS busiincome, indiincome+busiincome AS totalincome, SUM(invoiceprice) AS invoicetotal, totalincome-invoicetotal AS netincome, soldprice/invoiceprice AS priceratio,  FROM **VehicleSoldIndividua**l JOIN **VehicleSoldBusiness** JOIN **Vehicle**  WHERE **VehicleSoldIndividual**.vin=**Vehicle.**vin  AND **VehicleSoldBusiness**.vin=**Vehicle**.vin  AND year IS NOT NULL  AND month IS NOT NULL  ORDER BY sale\_year DESC, sale\_month DESC; |

* + Display the list on **Monthly Sales screen**
* Users are able to select one manufacturer’s name from the list:
  + Run **Drill-down** task with the manufacturer’s name

### 

### 

### 

### 

### **Monthly Sales Drill-Down**

Abstract Code

* User select a month/year from the list on the **Monthly Sales screen**
* Retrieve the **sale**s in select month/year group from the **Monthly Sales** task.
* Group the **sale** by **sale**.salesperson then calculates the total vehicles and total sales for each **sale**.salesperson. sort the Sale.salesperson by total vehicles descending followed by total sales descending. The first **sale**.salesperson is the top salesperson

|  |
| --- |
| SELECT count as totalvehicle, SUM(soldprice) AS totalsales  FROM  SELECT \* FROM **VehicleSOLDIndividual**  UNION  SELECT \* FROM **VehicleSOLDBusiness**  GROUP BY salespersonusername  ORDER BY totalvehicle DESC; |

* Display the top salesperson on the **Monthly Sales Drill-Down Screen**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.