Phase 2 Abstract Code w/SQL | CS 6400 – Fall 2023 | **Team 085**

Login

Abstract Code

- User clicks on the User Login Link
- User enter *password*($password) and *Username*($Username) into input fields
- User click "***Enter***" key:

  a. If form is validated, then

```
SELECT password FROM User WHERE username='$Username';
```

  1. If username does not have a record, then redisplay **login** form with error message indicating username not found.
  2. If username is found, but password does not match, then redisplay **login** form with error message indicating login credentials are not correct.
  3. Redirect to **Search page**

  b. Else redisplay **login** form with error message indicating which filed does not pass from validation.

Public Vehicle Search

Abstract Code

- View **Login** Page. Populate vehicle type, manufacturer, model year, fuel type, color DROPDOWNs and Keyword blank
- Read the list of *Manufacturers* from Vehicle Manufacturer table and populate the Manufacturer dropdown.

```
SELECT manufacturer_name FROM VehicleManufacturer;
```

- Read the list of Vehicle Types from Vehicle Type table and populated the Vehicle Type dropdown.

```
SELECT vehicle_type FROM VehicleType;
```

- For Vehicle Type, Manufacturer, Model Year, Fuel Type and Color dropdowns, allow user(s) to select desired value.
    a. Find and display Vehicle(s) matched by user selected value(s).

```
SELECT V.vin, V.vehicle_type, V.manufacturer_name, V.fuel_type,
V.model_name, V.model_year, V.description, V.mileage, VC.color

FROM Vehicle V LEFT JOIN VehicleColor VC ON V.vin = VC.vin
WHERE
   V.vehicle_type = '$vehicleType' AND
   V.manufacturer_name = '$ManufactureName' AND
   V.model_year = '$ModelYear' AND
   V.fuel_type = '$FuelType' AND
   VC.color = '$VehicleType';
```

   b. Find PartOrder.PurchaseOrderNumber that contains Vehicle.Vin**.**

```
SELECT purchase_order_number FROM PartOrder WHERE vin = '$Vin';
```

   c. Find Part.PartNumber, Part.Status that are in PartOrder.PurchaseOrderNumber.

```
SELECT Part.part_number, Part.status FROM Part JOIN PartOrder ON
Part.purchase_order_number= PartOrder.purchase_order_number WHERE ;
```

   d. If the logged in user's role is one of {Manager, Clerk, Owner}, then display all matched Vehicles regardless of Part.Status.

```
   SELECT DISTINCT Vehicle.vin, Vehicle.model_name, Vehicle.manufacturer_name
   FROM Vehicle
   JOIN PartOrder ON Vehicle.vin = PartOrder.vin
   JOIN Part ON PartOrder.purchase_order_number = Part.purchase_order_number;
```

   e. Else, display only matched Vehicles with all parts installed.

```
SELECT DISTINCT Vehicle.vin, Vehicle.model_name, Vehicle.manufacturer_name
FROM Vehicle
JOIN PartOrder ON Vehicle.vin = PartOrder.vin
JOIN Part ON PartOrder.purchase_order_number = Part.purchase_order_number
WHERE NOT EXISTS (
  SELECT 1
  FROM Part p2
  WHERE p2.purchase_order_number = PartOrder.purchase_order_number
  AND p2.status != 'Installed'
);
```

- For Keyword, user(s) can enter any text.
    a. Find and display Vehicle(s) whose Vehicle.Manufacturer**,** Vehicle.ModelYear**,** Vehicle.ModelName**, or** Vehicle.Description contains the keyword that user(s) entered.

```
SELECT vin, manufacturer_name, model_year, model_name, description

FROM Vehicle

WHERE

  manufacturer_name LIKE '%' || :keyword || '%' OR

  model_year::TEXT LIKE '%' || :keyword || '%' OR

  model_name LIKE '%' || :keyword || '%' OR

  description LIKE '%' || :keyword || '%';
```

    b. Find PartOrder.PurchaseOrderNumber that contains Vehicle.Vin.

```
SELECT purchase_order_number FROM PartOrder WHERE vin = '$Vin';
```

    c. Find Part.PartNumber, Part.Status that are in PartOrder.PurchaseOrderNumber.

```
SELECT Part.part_number, Part.status FROM Part JOIN PartOrder ON
Part.purchase_order_number= PartOrder.purchase_order_number WHERE ;
```

d. If the logged in user's role is one of {Manager, Clerk, Owner}, then display all matched Vehicles regardless of Part.Status.

```
  SELECT DISTINCT Vehicle.vin, Vehicle.model_name, Vehicle.manufacturer_name
  FROM Vehicle
  JOIN PartOrder ON Vehicle.vin = PartOrder.vin
  JOIN Part ON PartOrder.purchase_order_number =
Part.purchase_order_number;
```

e. Else, display only matched Vehicles with all parts installed.

```
  SELECT DISTINCT Vehicle.vin, Vehicle.model_name,
Vehicle.manufacturer_name
  FROM Vehicle
  JOIN PartOrder ON Vehicle.vin = PartOrder.vin
  JOIN Part ON PartOrder.purchase_order_number =
Part.purchase_order_number
  WHERE NOT EXISTS (
    SELECT 1
    FROM Part p2
    WHERE p2.purchase_order_number =
PartOrder.purchase_order_number
    AND p2.status != 'Installed'
  );
```

Add Vehicle

Abstract Code

- User will select ***Add Vehicle*** button or link and be navigated to **Add Vehicle** form
- Form is initialized with manufacturer_name from Vehicle Manufacturer

```
SELECT manufacturer_name
FROM vehiclemanufacturer;
```

and vehicle_type from Vehicle Types for drop downs

```
SELECT vehicle_type
FROM vehicletype;
```

- User will enter either driver's license or tax ID to customer search field
  a. Run **Find Customer** task;
  b. If customer exists

            i. Display customer name
    c. If customer does not exist
            i. Run **Add Customer** task
            ii. Display newly added customer name

- Form will gather all relevant Vehicle, Vehicle Manufacturer, and Vehicle Type details as well as *purchase_date*, *vehicle_condition*
- If all data entered is valid
    a. New vehicle will be inserted into table

```
WITH new_vin AS (
INSERT INTO vehicle (vin, vehicle_type, manufacturer_name, fuel_type, model_name, model_year,
description,mileage)
VALUES('$vin', '$vehicle_type', '$manufacturer_name', '$fuel_type', '$model_name', '$model_year',
'$description', '$mileage') RETURNING vin
)
INSERT INTO Sell (customer_id, vin, username, purchase_price, purchase_date, vehicle_condition)
VALUES ($CustomerID, (select vin from new_vin), $Username, $PurchasePrice, $PurchaseDate,
$VehicleCondition);
```

    b. User will be taken to the **detail page** for the vehicle

- Else
    a. Error message will display
    b. New vehicle will not be added

Add Customer

Abstract Code

- User enters customer's driver's license or tax ID into input fields
- If data is valid for either driver's license or tax ID fields, then:
    a. When ***Enter*** button is clicked

```
SELECT tax_id_number
FROM customerbusiness
WHERE customerbusiness.tax_id_number = '$tax_id_number';
```

Or

```
SELECT drivers_license_number
        FROM customerindividual
        WHERE customerindividual.drivers_license_number = '$drivers_license_number';
```

            i. If Customer record is not found

```
1. If customer is type of individual_customer
    1. Provide form for fields relevant to individual customer
    2. User enters new data
        1. If data is valid for individual_customer
            1. New customer record is added
```

```
INSERT INTO customer
VALUES (gen_random_uuid(), '$street','$city', '$state', '$postal_code', '$phone_number');

SELECT customer_id
FROM customer
WHERE street = '$street' and city='$city' and state='$state' and postal_code='$postal_code' and
phone_number='$phone_number';

INSERT INTO customerindividual (drivers_license_number, customer_id, first_name, last_name)
VALUES('$drivers_license_number', '$customer_id','$first_name', '$last_name');
```

```
            2. User is returned to Add Vehicle form
        2. Else
            1. New customer record is not added
            2. Error is displayed
2. Else if customer is type of business_customer
    1. Provide form for fields relevant to individual customer
    2. User enters new data
        1. If data is valid for business_customer
            1. New customer record is added
```

```
INSERT INTO customer
VALUES (gen_random_uuid(), '$street','$city', '$state', '$postal_code', '$phone_number');

SELECT $customer_id
FROM customer
WHERE street = '$street' and city='$city' and state='$state' and postal_code='$postal_code' and
phone_number='$phone_number';

INSERT INTO customerbusiness (tax_id_number, customer_id, contact_name, title)
VALUES('$tax_id_number', '$customer_id','$first_name', '$title');
```

```
            2. User is returned to Add Vehicle form
        2. Else
            1. New customer data is not added
            2. Error is displayed
```

Find Customer

Abstract Code

- User enters customer’s driver’s license or tax ID *i*nto input fields
- If data is valid for either driver’s license or tax ID fields, then:

a. When ***Search*** button is clicked; search by provided key on Customers

```
SELECT *
FROM customerbusiness
WHERE tax_id_number = '$tax_id_number';
```

Or

```
SELECT *
FROM customerindividual
WHERE drivers_license_number = '$drivers_license_number';
```

i. If customer record is found
   1. Returns relevant customer data

Add Sales

Abstract Code

- User selects the ***Sell car button in the*** **Vehicle Details** page
- Upon:
  a. Select *Buyer* input field
     i. Look up a Buyer using *Drivers License Number* or *Tax Identification number* using **Find Customer** task
        1. If Customer does not exist, link to **Add Customer** task
     ii. Select Customer and press ***Enter*** button
- Enter *Sales date* input field
- Click ***Save*** or ***Enter*** button

```
INSERT INTO buy (customer_id, vin, username, sale_date) VALUES ($CustomerID, $VIN, $Username,
$SaleDate);
```

Add Parts

Abstract Code

- User opens a **Vehicle Details** page and clicks the ***Add Part Order*** button to open **Add Parts Order** form.
- In Part section:
  a. Enter *Part Number*, *Description*, *purchase order*, *cost, quantity* and *status* input fields.
  b. User must select Vendor - Jump to the **Search and add vendors** task
  c. Select ***Save*** button in parts section – Stored in application memory
- If more parts must be added, repeat steps in the Part section.
- When done, return to **Add Parts Order** form

Add Parts Order

Abstract Code

- User opens a **Vehicle Details** page and selects a Vehicle using the **Vehicle details Task**
- Then, click the ***Add Part Order*** button to open **Add Parts Order** form.
- Enter parts using the **Add Parts Task**
- Select or add vendors using the **Search and add vendors Task**
- Compute $TotalCost from Parts saved in application memory. Add up cost of each Part
- Press Save

```
WITH max_count AS (
  select TO_CHAR(count(*)+1, 'fm00') count from PartOrder where vin = $VIN
) INSERT INTO PartOrder (
  purchase_order_number, vin, part_vendor_name, username, total_cost
) VALUES (
 CONCAT($VIN, ‘-’, (select count from max_count)), $VIN, $VendorName, $Username, $TotalCost);
```

- Return to **Vehicle Details** page or display error if needed.

Search and add vendors

Abstract Code

- User focuses on Parts section in **Add Part Order** form
- To choose existing Vendor:
  a. User selects the ***Search*** link to open **Vendor Search** form
  b. Find the Vendor using *Name* or *Phonenumber.* Store in $VendorName

```
SELECT name FROM PartVendor WHERE phone_number = $Phonenumber OR name = $Name;
```

  c. Select Vendor from list of result and press ***Enter*** button
- To add new Vendor:
  a. User select the ***Add Vendor*** link to open the **Add Vendor** form
  b. Enter *name*, *address* and *phonenumber* in input fields
  c. Press ***Enter*** button

```
INSERT INTO PartVendor (name, phone_number, street, city, state, postal_code) VALUES ($Name,
$Phonenumber, $Street, $City, $State, $PostalCode);
```

- Return to previous **Part** section with Vendor selected in **Parts Order** form

Vehicle details

Abstract Code:

- User **Login**
- User searches for Vehicle using **Search Vehicle** task
- If User is a public search:

- Display **vehicle details**: Show the following information for the selected vehicle: Vehicle type, Manufacturer name, Model name, Model year, Fuel type, Color(s), Mileage, Description.

```
SELECT v.vehicle_type, v.manufacturer_name, v.model_name, v.model_year, v.fuel_type, vc.color,
v.mileage, v.description
FROM Vehicle v
JOIN VehicleColor vc ON v.vin = vc.vin
WHERE v.vin = '$VehicleVIN';
```

- If user already logged in, check the roles in the User table:

a. If User is verified to have "Salespeople" role:

i. Display **vehicle details**: Show the following information for the selected vehicle: Vehicle VIN, Vehicle type, Manufacturer name, Model name, Model year, Fuel type, Color(s), Mileage, Description.

```
SELECT v.vin, v.vehicle_type, v.manufacturer_name, v.model_name, v.model_year, v.fuel_type,
vc.color, v.mileage, v.description
FROM Vehicle v
JOIN VehicleColor vc ON v.vin = vc.vin
WHERE v.vin = '$VehicleVIN';
```

b. If User is verified to have "Inventory Clerks" role:

i. Display **vehicle details**: Show the following information for the selected vehicle: Vehicle VIN, Vehicle type, Manufacturer name, Model name, Model year, Fuel type, Color(s), Mileage, Description.

ii. Show the seller's contact information, the original purchase price, the purchase date.

```
SELECT v.vin, v.vehicle_type, v.manufacturer_name, v.model_name, v.model_year, v.fuel_type,
vc.color, v.mileage, v.description, c.phone_number, s.purchase_price, s.purchase_date
FROM Vehicle v
JOIN VehicleColor vc ON v.vin = vc.vin
JOIN Sell s ON v.vin = s.vin
JOIN Customer c ON s.customer_id = c.customer_id
WHERE v.vin = '$VehicleVIN';
```

iii. Show all **parts orders** for the **selected vehicle**, sorted by order number (e.g., 123-01, 123-02). Include information such as the order number, vendor name, orand total cost. Allow user to click on **a specific parts order** for details, jump to Vehicle parts order

```
SELECT p.purchase_order_number, p.part_vendor_name, p.total_cost
FROM PartOrder p
WHERE p.vin = '$VehicleVIN'
ORDER BY p.purchase_order_number;
```

iv. Calculate and show **the total cost of all parts orders.**

```
SELECT SUM(total_cost)
FROM PartOrder p
WHERE p.vin = '$VehicleVIN';
```

c. If User is verified to have "Manager" role:

i. Display all the details same as "Inventory Clerks" role

ii. If the car has been sold, display the buyer's contact information (everything except their driver's license or tax ID number), sales date, and the salesperson's name (first and last) .

```
SELECT v.vin, CONCAT_WS(' ',ci.first_name, ci.last_name) AS contact_info, c.street, c.city, c.state,
c.postal_code, c.phone_number, b.username
FROM Vehicle v
JOIN Buy b ON v.vin = b.vin
JOIN Customer c ON b.customer_id = c.customer_id
JOIN CustomerIndividual ci ON ci.customer_id = c.customer_id
WHERE v.vin = '$VehicleVIN'
UNION
SELECT v.vin, CONCAT_WS(' ',cb.contact_name, cb.title) AS contact_info, c.street, c.city, c.state,
c.postal_code, c.phone_number, b.username
FROM Vehicle v
JOIN Buy b ON v.vin = b.vin
JOIN Customer c ON b.customer_id = c.customer_id
JOIN CustomerBusiness cb ON cb.customer_id = c.customer_id
WHERE v.vin = '$VehicleVIN';
```

d. If User is verified to have "Owner" role:

i. Display all the details same as "Manager" role

Vehicle parts order

Abstract Code:

- User in **Vehicle Details** page, user click on **a specific parts order** for details
- Display details of the selected parts order:

a. Show information related to the selected parts order, including:

i. Order number (e.g., 123-01).

ii. Vendor details (name, address, phone number).

iii. List of parts included in the order:

iv. For each part, display its description, part number, cost, quantity, and current status (ordered, received, installed).

b. Calculate and display the **Total Cost of the part order:** Sum the costs of all parts in the selected parts order.

```
SELECT po.purchase_order_number, po.part_vendor_name, pv.street, pv.city, pv.state,
pv.postal_code, pv.phone_number, p.part_number, p.description, p.cost, p.quantity, p.status,
po.total_cost
FROM partorder po
JOIN partvendor pv ON po.part_vendor_name = pv.name
LEFT JOIN part p ON p.purchase_order_number = po.purchase_order_number
WHERE po.purchase_order_number = '$OrderNumber';
```

- Allow user to click on **Update status** for each part, jump to **Update part status** task
- Allow user to click **Add part order** button, jump to Add Parts Order Form

- User click on ***Back***: Go back to the list of parts orders (Vehicle Details) or select another parts order for viewing.

Update part status

Abstract Code:

- User in View **vehicle parts order**
- Click on **Update status** for one part to update the part order status.
- Show the current status of the part (ordered, received, installed). Check status:

a. If the current status is "**installed"**, disable the update option.

b. If the current status is "**received**", allow to select status "**installed**" : Update the part status to "**installed**."

c. If the current status is "**ordered**", allow to select status "**received**" or "**installed**": Update the part status to "**received**" or "**installed".**

```
UPDATE part
SET status = '$PartStatus'
WHERE part_number = '$PartNumber';
```

- When **Save** button is clicked, save status:
- Return to view **vehicle part order.**

Seller History

Abstract Code

- User **Logs In.**
- User is verified to have either 'owner' or 'manager' role in the User table.

```
select
  ci.first_name || ' ' || ci.last_name as seller_name,
  count(s.vin) as number_of_vehicles_sold,
  avg(s.purchase_price) as average_purchase_price,
  sum(p.quantity) / count(s.vin) as average_number_of_parts_per_vehicle,
  sum(p.cost * p.quantity) / count(s.vin) as average_cost_of_parts_per_vehicle
from sell s
inner join customerindividual ci on ci.customer_id = s.customer_id
left join partorder po on po.vin = s.vin
left join customer c on c.customer_id = s.customer_id
left join part p on p.purchase_order_number = po.purchase_order_number
group by seller_name
union
select
  cb.tax_id_number as seller_name,
  count(s.vin) as number_of_vehicles_sold,
  avg(s.purchase_price) as average_purchase_price,
  sum(p.quantity) / count(s.vin) as average_number_of_parts_per_vehicle,
  sum(p.cost * p.quantity) / count(s.vin) as average_cost_of_parts_per_vehicle
from sell s
inner join customerbusiness cb on cb.customer_id = s.customer_id
left join partorder po on po.vin = s.vin
left join customer c on c.customer_id = s.customer_id
left join part p on p.purchase_order_number = po.purchase_order_number
group by seller_name;
```

Average Time in Inventory

Abstract Code

- User **Logs In.**
- User is verified to have either 'owner' or 'manager' role in the User table.

```
select
  vt.vehicle_type,
  avg(date_part('Day', s.purchase_date::timestamp - b.sale_date::timestamp) + 1) as
average_days_in_inventory
from vehicletype vt
left join vehicle v on v.vehicle_type = vt.vehicle_type
left join buy b on b.vin = v.vin
left join sell s on s.vin = v.vin
group by vt.vehicle_type;
```

Price Per Condition

Abstract Code

- User **Logs In.**
- User is verified to have either 'owner' or 'manager' role in the User table.

```
with all_vehicle_conditions as (
  select cond as vehicle_condition
  from (values ('Excellent'), ('Very Good'), ('Good'), ('Fair')) t(cond)
),
cond_and_type as (
  select
    vt.vehicle_type,
    avc.vehicle_condition
  from all_vehicle_conditions avc
  cross join vehicletype vt
),
augmented_sell as (
  select
    s.vin,
    s.purchase_price,
    s.vehicle_condition,
    v.vehicle_type
  from sell s
  natural join vehicle v
)
select
  cond_and_type.vehicle_type,
  cond_and_type.vehicle_condition,
  avg(augmented_sell.purchase_price) as avg_purchase_price
from cond_and_type
left join augmented_sell
on cond_and_type.vehicle_type = augmented_sell.vehicle_type and
   cond_and_type.vehicle_condition = augmented_sell.vehicle_condition
group by
  cond_and_type.vehicle_type,
  cond_and_type.vehicle_condition;
```

Parts Statistics

Abstract Code

- User **Logs In.**
- User is verified to have either 'owner' or 'manager' role in the User table.

```
select
  pv.name,
  sum(quantity) as total_number_of_parts,
  sum(total_cost) as total_dollar_amount
from partorder po
join partvendor pv on pv.name = po.part_vendor_name
natural join part p
group by pv.name;
```

Monthly Sales Summary

Abstract Code

- User **Logs In.**
- User is verified to have either 'owner' or 'manager' role in the User table.

```
with total_parts_cost_per_vehicle as (
  select
    buy.vin,
    sum(partorder.total_cost) * 1.10 as total_po_cost
  from buy
  natural join partorder
  group by buy.vin
),
purchase_price_per_vehicle as (
  select
    date_part('year', buy.sale_date::date)::varchar as sale_year,
    date_part('month', buy.sale_date::date)::varchar as sale_month,
    buy.vin,
    sell.purchase_price
  from buy
  natural join sell
)
select
  pp.sale_year,
  pp.sale_month,
  count(pp.vin) as total_number_of_vehicles_sold,
  sum(pp.purchase_price) * 1.25 + sum(tc.total_po_cost) * 1.10 as total_sales_income,
  sum(pp.purchase_price) * 0.25 + sum(tc.total_po_cost) * 0.10 as total_net_income
from purchase_price_per_vehicle pp
natural join total_parts_cost_per_vehicle tc
group by pp.sale_year, pp.sale_month
order by pp.sale_year desc, pp.sale_month desc;
```

Monthly Sales Drilldown

Abstract Code

- User **Logs In.**
- User is verified to have either ‘owner’ or ‘manager’ role in the User table.

```
with total_parts_cost_per_vehicle as (
  select
    buy.vin,
    sum(partorder.total_cost) * 1.10 as total_po_cost
  from buy
  natural join partorder
  group by buy.vin
)
select
  b.username,
  sp.first_name || ' ' || sp.last_name as sales_people,
  count(b.vin) as number_of_vehicles_sold,
  sum(s.purchase_price * 1.25) + sum(tc.total_po_cost) * 1.10 as sales_income
from buy b
natural join salespeople sp
natural join sell s
natural join total_parts_cost_per_vehicle tc
where
  date_part('year', b.sale_date::date)::varchar = '$year' and
  date_part('month', b.sale_date::date)::varchar = '$month'
group by b.username, sales_people
order by number_of_vehicles_sold desc, sales_income desc;
```