Phase 2 Report | CS 6400 - Fall 2017 | Team 091

Table of Contents

Abstract Code

Login

Abstract Code

- User enters *username*($Username), *password*($Password) into input fields
- If ***Customer*** radio button was selected:
    - When ***Sign in*** button is clicked:

$Employer_Number = SELECT employer_number FROM `User` INNER JOIN `Clerk` ON `User`.email = `Clerk`.email WHERE `User`.username = '$Username';
$Record_Username = SELECT username FROM `User` INNER JOIN `Customer` ON `User`.email = `Customer`.email WHERE `User`.username = '$Username';
$Record_Password = SELECT password FROM `User` INNER JOIN `Customer` ON `User`.email = `Customer`.email WHERE `User`.username = '$Username';

        - If $Employer_Number is not NULL, then error message would be displayed to explain Clerk user could not login as Customer user.
        - If $Record_Username is NULL, then go to **Registration** form.
        - Else if $Record_Password is equal to $Password, then go to **Customer Main** form.
        - Else an error message would be displayed to inform the user that the inputted password was incorrect.

- If ***Clerk*** radio button was selected:
    - When ***Sign in*** button is clicked:

$Email = SELECT email FROM `User` INNER JOIN `Customer` ON `User`.email = `Customer`.email WHERE `User`.username = '$Username';
$Temp_Password = SELECT temp-password FROM `User` INNER JOIN `Clerk` ON `User`.email = `Clerk`.email WHERE `User`.username = '$Username';
$Record_Username = SELECT username FROM `User` INNER JOIN `Clerk` ON `User`.email = `Clerk`.email WHERE `User`.username = '$Username';
$Record_Password = SELECT password FROM `User` INNER JOIN `Clerk` ON `User`.email = `Clerk`.email WHERE `User`.username = '$Username';

- If $Email is not NULL, then error message would be displayed to explain Customer user could not login as Clerk user.
- If $Record_Password is equal to $Password and $Record_Username is equal to $Username, then go to **Clerk Main** form.
- If $Temp_Password is equal to $Password and $Record_Username is equal to $Username, then prompt to reset password by entering new password twice.
- If $Temp_Password is not equal to $Password and $Record_Password is not equal to $Password and $Record_Username is equal to $Username, an error message would be displayed to inform the user that the inputted password was incorrect.

Main Menu / Navigation Bar

Abstract Code

If user login as Customer:

- Show ***"View Profile"***, ***"Check Tool Availability"***, ***"Make Reservation"***, ***"Purchase Tool"*** and ***"Exit"*** tabs.
- Upon:
  - Click ***View Profile*** button- Jump to the **View Profile** task.
  - Click ***Check Tool Availability*** button- Jump to the **Check Tool Availability** task.
  - Click ***Make Reservation*** button- Jump to the **Make Reservation** task.
  - Click ***Purchase Tool*** button- Jump to the **Purchase Tool** task.
  - Click ***Exit*** button- exit main menu and go back to the **Login** form.

If user login as Clerk:

- Show ***"Pick-Up Reservation"***, ***"Drop-Off Reservation"***, ***"Add New Tool"***, ***"Service Order"***, ***"Service Status"***, ***"Sell Tool"***, ***"Sale Status"***, ***"Generate Reports"*** and ***"Logout"*** tabs.
- Upon:
    - Click ***Pick-Up Reservation*** button- Jump to the **Pick-Up Reservation** task.
    - Click ***Drop-Off Reservation*** button- Jump to the **Drop-Off Reservation** task.
    - Click ***Add New Tool*** button- Jump to the **Add New Tool** task.
    - Click ***Service Order*** button- Jump to the **Service Order** task.
    - Click ***Service Status*** button- Jump to the **Service Status** task.
    - Click ***Sell Tool*** button- Jump to the **Sell Tool** task.
    - Click ***Sale Status*** button- Jump to the **Sale Status** task.
    - Click ***Generate Reports*** button- Jump to the **Generate Reports** task.
    - Click ***Log Out*** button- Invalidate login session and go back to the **Login** form.

Registration

Abstract Code

Customer opens **Customer Registration Form**:

- User enters *username*($Username), *password*($Password), *re-type password*($Password2), *e-mail address*($Email), *first name*($First_Name), *middle name*($Middle_Name), *last name*($Last_Name), *home phone*, *work phone* and *cell phone* information.

```
$Record_Username = SELECT username FROM `User` INNER JOIN `Customer` ON `User`.email = `Customer`.email WHERE `User`.username = '$Username';
```

- If $Record_Username is not NULL, then display “Error: Customer already exists”.
- Front-end scripts parse phone numbers to area code($Area_Code), number($Phone_Number), extension($Phone_Extension) and phone type($Phone_Type).
- Front-end scripts check if primary phone number was declared by Customer($Is_Primary).
- User enters *street address*($Street_Address), *city*($City), *state*($State), *9-digit zip code*($Zipcode) with hyphen.
- User enters *credit card number*($CC_Number) and *name on card*($CC_Name).
- User selects *credit card expiration month*($CC_Month) and *credit card expiration year*($CC_Year) from dropdown menu.

- User enters *credit card CVC 3-digit number*($CC_CVC).
- When ***Register*** button is clicked:
    - If $Password is not equal to $Password2, then the user has to re-enter the passwords and click the ***Register*** button again.
    - Else if all required fields were filled, then integrate the new user information into the database and return to **Login** form.

```
INSERT INTO `User` (email, first_name, middle_name, last_name, username, password)
VALUE ('$Email', '$First_Name', '$Middle_Name', '$Last_Name', '$Username', '$Password');
```

```
INSERT INTO `Customer` (email, add_street, add_city, add_state, add_zip_code,
cc_number, cc_name_on_card, cc_expiration_month, cc_expiration_year, cc_cvc) VALUE
('$Email', '$Street_Address', '$City', '$State', $Zipcode, $CC_Number, '$CC_Name',
$CC_Month, $CC_Year, $CC_CVC);
```

```
INSERT INTO `CustomerPhoneNumber` (email, type, area_code, phone_number,
extension, primary) VALUE ('$Email', $Phone_Type, $Area_Code, $Phone_Number,
$Phone_Extension, $Is_Primary);
```

    - Else display “Error: missing required value(s)” and mark the missing value. The user has to enter the blank field(s) and and click the ***Register*** button again.

View Profile

Abstract Code

Customer clicks on ***View Profile*** button from **Main Menu**:

- Run the **View Profile** task: extracting information about the rental customer and their profile where $Username is the identifier of the current Customer using the system from the HTTP Session/Cookie.
- Find the current Customer using the $Username; Display user’s first and last name; Display user’s E-mail; Display user’s all phone numbers; Display user’s Address.

```
SELECT
    `User`.first_name
    `User`.last_name
    `User`.email
    `Customer`.add_street
    `Customer`.add_city
    `Customer`.add_state
    `Customer`.add_zip_code
    `CustomerPhoneNumber`.type
```

```
        `CustomerPhoneNumber`.area_code
        `CustomerPhoneNumber`.phone_number
        `CustomerPhoneNumber`.extension
FROM `User`
INNER JOIN `Customer`
        ON `User`.email = Customer.email
INNER JOIN `CustomerPhoneNumber`
        ON `Customer`.email = `CustomerPhoneNumber`.email
WHERE `User`.username = '$Username';
```

- Find each reservation made by the current Customer:
    - Display all tools reserved in this single reservation.
    - Display reservation start date and end date.
    - Display the names of Clerk who handled the tools pick-up / drop-off sessions.
    - Display the number of reservation days.
    - Display the total deposit price and total rental price.

```
SELECT
        `Reservation`.reservationID
        `Reservation`.pick_up_date
        `Reservation`.drop_off_date
        `Reservation`.start_date
        `Reservation`.end_date
        DropoffClerk.first_name AS d_first_name
        DropoffClerk.last_name AS d_last_name
        PickupClerk.first_name AS p_first_name
        PickupClerk.last_name AS p_first_name
        `Tool`.short_desc
        `Tool`.original_price
FROM `User`
INNER JOIN `Customer`
        ON `User`.email = `Customer`.email
INNER JOIN `Reservation`
        ON `Customer`.email = `Reservation`.renter-email
INNER JOIN `User` as `DropoffClerk`
        ON DropoffClerk.email = `Reservation`.drop_off_email
INNER JOIN `User` as `PickupClerk`
        ON PickupClerk.email = `Reservation`.pick_up_email
INNER JOIN `Renting`
        ON `Renting`.reservationID = `Reservation`.reservationID
INNER JOIN `Tool`
        ON `Tool`.tool_number = `Renting`.tool_number
WHERE `User`.username = '$Username' ;
```

- When ready, user selects next action from choices in **Main Menu.**

Check Tool Availability

Abstract Code

- Customer selects Tool or inputs tool information.
- The Tool information is checked against the list of tools that are Reserved.

```
SELECT
        tool_number,
        tool_type,
       sub_type,
       sub_option,
       short_desc,
       material,
       length
FROM `Tool`
INNER JOIN `Reservation`
    ON `Tool`.sub_option = `Reservation`.sub_option
    AND `Tool`.sub_type = `Reservation`.sub_type
    AND `Tool`.tool_type = `Reservation`.tool_type;
```

Make Reservation

Abstract Code

- Update Reservation with the Data of the tool that should be reserved and the user ID

```
UPDATE `Reservation`
SET email='$UserID', tool_type='$ToolType', StartDate='$StartDate', EndDate='$EndDate';
```

Purchase Tool

Abstract Code

- Update the Sale Order table with the SaleDate of when the tool was sold
- Delete the Tool from the Inventory

```
UPDATE `SaleOrder`
SET sale_date = $SaleDate
WHERE `SaleOrder`.sale_order_id = $SaleOrderId
DELETE FROM `Tool`
WHERE `Tool`.tool_number = $ToolNumber;
```

Pick-up Reservation

Abstract Code

- Search Reservation for all reservations waiting to be picked up
- Find start-date and end-date in Reservation for each reservation to be picked up
- Find username and customer-id in Customer for each reservation to be picked up
- Display reservation-id, customer, customer-id, start-date and end-date for each reservation to be picked up

(all of above accomplished with following:)

```
SELECT
        `User`.email,
        `User`.username,
        `Reservation`.reservationID,
        `Reservation`.renter-email,
        `Reservation`.start_date,
        `Reservation`.end_date,
        `Reservation`.pickup_date
FROM `User`,
INNER JOIN `Reservation`
```

```
        ON `Reservation`.renter_email=`User`.email
WHERE `Reservation`.pick_up_date is NULL;
ORDER BY `Reservation`.reservationID;
```

- Upon click reservation-id
  - Find full-name in User

```
SELECT
        `User`.first_name,
        `User`.last_name,
        `User`.email,
        `Reservation`.renter_email,
        `Reservation`.reservationID
FROM `User`, `Reservation`
INNER JOIN `Reservation`
        ON `Reservation`.renter_email=`User`.email
WHERE `Reservation`.reservationID=$ResID;
```

  - For each tool on reservation, find original-price in Tool

```
SELECT
        `Reservation`.tool_number,
        `Reservation`.reservationID,
        `Tool`.original_price,
        `Tool`.tool_number
FROM `Reservation`, `Tool`
INNERJOIN `Tool`
        ON `Reservation`.tool_number=`Tool`.tool_number
WHERE `Reservation`.reservationID=$ResID;
```

  - Calculate total deposit and rental price for each tool
  - Calculate total deposit price and total rental price for reservation-id
  - Display pop-out detail with reservation-id, full-name-total deposit and total-rental price
- Upon entering reservation id and click ***PickUp***:
  - Display reservation summary
    - If credit card info in summary NULL and Click ***Confirm PickUp***
      - Find credit-card in Customer

```
SELECT
        `Customer`.cc_number,
        `Customer`.cc_name_on_card,
```

```
        `Customer`.cc_expiration_month,
        `Customer`.cc_expiration_year,
        `Customer`.cc_cvc
FROM `Customer`
WHERE email IN (SELECT `Reservation`.email
                        FROM `Reservation`
                        WHERE `Reservation`.email=$ResID);
```

- Change credit card in Customer with credit-card information

```
INSERT INTO `Customer` (cc_number, cc_name_on_card, cc_expiration_month,
cc_expiration_year, cc_cvc)
VALUES ($cc_number, '$cc_name_on_card', $cc_expiration_month, $cc_expiration_year,
$cc_cvc)
```

- Update pick-up-date in Reservation

```
UPDATE `Reservation`
SET pick_up_date=$CurrentDate
WHERE reservationID=$ResID;
```

- Display Rental Contract
  - If click ***Print Contract***--print contract

- If credit card info in summary Not Null and click Confirm PickUp
  - Update credit-card in Customer with information from form

```
UPDATE `Customer`
SET
        cc_number=$NewCCNum,
        cc_name_on_card='$NewNameCard',
        cc_expiration_month=$NewExpMonth,
        cc_expiration_year=$NewExpYear,
        cc_cvc=$NewCVC
WHERE email IN (SELECT `Reservation`.email
                        FROM `Reservation`
                        WHERE `Reservation`.email=$ResID);
```

- Charge credit card with credit-card information
- Update pick-up date in Reservation

```
UPDATE `Reservation`
SET pick_up_date=$CurrentDate
WHERE reservationID=$ResID;
```

- Display Rental Contract
  - If click ***Print Contract***--print contract

Drop-off Reservation

Abstract Code

- Search Reservation for all reservations that are ready for return (have been picked up but not returned)
- Find start-date and end-date in Reservation for each reservation to be returned
- Find username and customer-id in Customer for each reservation to be returned
- Display reservation-id, customer, customer-id, start-date and end-date for each reservation to be returned

(all of above accomplished with following:)

```
SELECT
    `User`.email,
    `User`.username,
    `Reservation`.reservationID,
    `Reservation`.renter_email,
    `Reservation`.start_date,
    `Reservation`.end_date,
    `Reservation`.drop_off_date
FROM `User`, `Reservation`
INNERJOIN `Reservation`
    ON `Reservation`.renter_email=`User`.email
WHERE `Reservation`.drop_off_date is NULL
ORDER BY `Reservation`.reservationID;
```

- Upon click reservation-id
  - Find full-name in User

```
SELECT
    `User`.first_name,
    `User`.last_name,
    `User`.email,
```

```
        `Reservation`.renter_email,
        `Reservation`.reservationID
FROM `User`, `Reservation`
INNERJOIN `Reservation`
        ON `Reservation`.renter_email=`User`.email
WHERE `Reservation`.reservationID=$ResID;
```

- For each tool on reservation, find original-price in Tool

```
SELECT
        `Reservation`.tool_number,
        `Reservation`.reservationID,
        `Tool`.original_price,
        `Tool`.tool_number
FROM `Reservation`, `Tool`
INNERJOIN `Tool`
        ON `Reservation`.tool_number=`Tool`.tool_number
WHERE `Reservation`.reservationID=$ResID;
```

- Calculate total deposit and rental price for each tool
- Calculate total deposit price and total rental price for reservation-id
- Display pop-out detail with reservation-id, full-name-total deposit and total-rental price

- Upon entering reservation id and click ***DropOff***:
  - Find full-name in User

```
SELECT
        `User`.first_name,
        `User`.last_name,
        `User`.email,
        `Reservation`.renter_email,
        `Reservation`.reservationID
FROM `User`, `Reservation`
INNERJOIN `Reservation`
        ON `Reservation`.renter_email=`User`.email
WHERE `Reservation`.reservationID=$ResID;
```

- For each tool on reservation, find original-price in Tool

```
SELECT
        `Reservation`.tool_number,
        `Reservation`.reservationID,
```

```
        `Tool`.original_price,
        `Tool`.tool_number
FROM `Reservation`, `Tool`
INNERJOIN `Tool`
        ON `Reservation`.tool_number=`Tool`.tool_number
WHERE `Reservation`.reservationID=$ResID;
```

- Calculate total deposit and rental price for each tool
- Calculate total deposit price, total rental price and total due for reservation-id
- Display Drop off Reservation with reservation-id, full-name-total deposit, total-rental price, total due
    - If click ***Drop Off***
        - Update drop-off-date in Reservation

```
UPDATE `Reservation`
SET drop_off_date=$CurrentDate
WHERE reservationID=$ResID;
```

- Display Final Receipt
    - If click ***Print Receipt***-print contract

Add Tool

Abstract Code

- On click of ***Type [radio button]*** get options for Sub-Type field from Tool

```
SELECT DISTINCT `Tool`.sub-option
FROM `Tool`
WHERE `Tool`.sub-type='$SelectedSubType';
```

- If Type is Power Tool, display Power Tool suboption fields
    - Dynamically determine options from Power

```
SELECT DISTINCT `Power_Tool`.sub_option
FROM `Power_Tool`;
```

- Dynamically determine accessory options from Accessory

```
SELECT DISTINCT `Tool_Accessory`.accessory_name
FROM `Tool_Accessory`;
```

    - If Type is Cordless, display cordless suboptions
        - Dynamically determine options from ~~Cordless~~ Power Source

```
SELECT DISTINCT 'Power Source'.battery_type
FROM 'Power Source';
```

- Upon selection of Sub-Type, get options for Sub-Option field from Tool

```
SELECT DISTINCT `Tool`.sub_option
FROM `Tool`
WHERE `Tool`.sub_type='$SelectedSubType';
```

- Upon click of Confirm
    - If width or length is feet, convert to inches
    - If amp, volt or power is “milli” or “kilo”, convert to decimal
    - Add tool to Tool
        - Add generic tool:(tool set to auto increment)

```
INSERT INTO 'Tool" (tool_number, tool_type, sub_type, sub_option, material, manufacturer,
original_price, length, width_diameter, short_desc)
VALUE ($ToolNumber, '$ToolType', '$SubType, '$SubOption, '$Material', '$Manufacturer',
$OriginalPrice, $Length, $WidthDiameter, '$ShortDesc')
```

        - Add type-specific information:

Example, if Hand Screwdriver:

```
INSERT INTO `Hand Screwdriver`(tool_number, screw_size)
VALUE ($ToolNumber, $ScrewSize)
```

Repair Tool

Abstract Code

- Upon click ***Search***
    - Get tool-number, short-desc (aggregate), original-price, from Tool that matches keyword in

Determine tool number from keyword search

```
SELECT
`Tool`.tool_number
FROM *
WHERE * LIKE '$Keyword'
```

Get information for tool that matches criteria with information for short description

```
SELECT
	`Tool`.tool_number,
	`Tool`.short_desc
	`Tool`.original_price
FROM `Tool`
WHERE `Tool`.tool_number=$ToolNumber
```

    - Calculate Rental Price and Deposit Price
    - Display list
- Upon click ***Type [radio button], Power Source, Sub-Type***
    - Get tool-number, short-desc, original-price from Tool that matches search criteria

Determine tool number matching criteria

```
SELECT
	`Tool`.tool_number
FROM `Tool`,`Power Source’
WHERE `Tool`.type='$Type' AND `PowerSource`.power_source='$PowerSource'
AND `Tool`.sub_type='$SubType'
```

Get information for tool that matches criteria with information for short description

```
SELECT
	`Tool.tool_number,
	`PowerSource`.power_source
	`Tool`.sub_option
	`Tool`.sub_type
	`Tool`.original_price
FROM `Tool`, `Power Source`
WHERE `Tool`.tool_number=$ToolNumber
```

    - Calculate Rental Price and Deposit Price
    - Display list
- Upon click ***Service Tool***

    - Load associated tool-number into “Tool ID” field
- Upon click ***Confirm***
    - If “Tool ID” field is Null, error “No tool ID”
    - If “Enter Service Cost” is Null, error “Enter repair cost”
    - Verify “Start Date” and “End Date” fields valid
    - Create Service-order
        - Add new record in Service_Order (Service Order set to auto increment)

```
INSERT INTO `ServiceOrder` (email, service-order-id, service-cost, service-start-date) VALUES ('$Email',$ServiceOrderID,$ServiceCost,'$StartDate');
```

View Service Status

NOTE: Aliases are used in this section to keep the queries short.

Abstract Code

- Search ServiceOrder for active records and return service-id, service-cost, service-start-date, service-end-date, service-tool-number and service-clerk-id. Format dates as datetime and repair-cost as $X.XX.
- Using sale-tool-number from ServiceOrder, read Tool for sub-type and short-desc, filtering results that match the selected sub-type in ***Type [radio button]***.
- Using service-clerk-id from ServiceOrder, read username from User for the matching user record.

```
SELECT so.serviceOrderID, '$' + CAST(CAST(so.service_cost as DECIMAL(10,2)) as VARCHAR(16)), DATE_FORMAT(so.service_start_date, '%Y-%m-%d %H:%i:%s'), DATE_FORMAT(so.service_end_date, '%Y-%m-%d %H:%i:%s'), so.tool_number, u.username, t.short_desc FROM `ServiceOrder` so INNER JOIN `User` u ON so.clerk_email = u.email INNER JOIN `Tool` t ON so.tool_number = t.tool_number WHERE so.service_start_date > NOW() AND so.service_end_date < NOW();
```

- If the ***Custom Search [input field]*** is not empty, search short-desc (aggregate), service-start-date, service-end-date, service-cost and username for matching values.

```
SELECT * FROM (SELECT so.serviceOrderID, '$' + CAST(CAST(so.service_cost as DECIMAL(10,2)) as VARCHAR(16)) as service_cost, DATE_FORMAT(so.service_start_date, '%Y-%m-%d %H:%i:%s') as service_start_date, DATE_FORMAT(so.service_end_date, '%Y-%m-%d %H:%i:%s') as service_end_date, so.tool_number, u.username, t.short_desc FROM `ServiceOrder` so INNER JOIN `User` u ON so.clerk_email = u.email INNER JOIN `Tool` t ON so.tool_number = t.tool_number WHERE so.service_start_date > NOW() AND so.service_end_date < NOW()) ex WHERE ex.tool_number LIKE '%$SearchTerm%' OR ex.short-desc LIKE '%$SearchTerm%' OR ex.service_start_date LIKE '%$SearchTerm%' OR ex.service_end_date LIKE '%$SearchTerm%' OR ex.service_cost LIKE '%$SearchTerm%' OR ex.username LIKE '%$SearchTerm%';
```

- If a ***Column Header [label]*** has been selected, order the results by the corresponding field for the selected column header, either ascending or descending

```
SELECT so.serviceOrderID, '$' + CAST(CAST(so.service_cost as DECIMAL(10,2)) as VARCHAR(16)), DATE_FORMAT(so.service_start_date, '%Y-%m-%d %H:%i:%s'), DATE_FORMAT(so.service_end_date, '%Y-%m-%d %H:%i:%s'), so.tool_number, u.username, t.short_desc FROM `ServiceOrder` so INNER JOIN `User` u ON so.clerk_email = u.email INNER JOIN `Tool` t ON so.tool_number = t.tool_number WHERE
```

```
so.service_start_date > NOW() AND so.service_end_date < NOW();
```

- Upon click ***Type [radio button]***
    - Search Tool for matching tool-type values
    - Read tool-number, short-desc (aggregate) from Tool
    - Search ServiceOrder for service-tool-number equal to tool-number
    - Read service-id, service-start-date, service-end-date, service-cost, service-clerk-id from ServiceOrder
    - Format service-start-date and service-end-date as datetime and service-cost as $X.XX.
    - Search User for user-id equal to service-clerk-id
    - Read username from User

```
SELECT so.serviceOrderID, '$' + CAST(CAST(so.service_cost as DECIMAL(10,2)) as VARCHAR(16)), DATE_FORMAT(so.service_start_date, '%Y-%m-%d %H:%i:%s'), DATE_FORMAT(so.service_end_date, '%Y-%m-%d %H:%i:%s'), so.tool_number, u.username, t.short_desc FROM `ServiceOrder` so INNER JOIN `User` u ON so.clerk_email = u.email INNER JOIN `Tool` t ON so.tool_number = t.tool_number WHERE so.service_start_date > NOW() AND so.service_end_date < NOW() ORDER BY $HeaderLabel;
```

- Upon click ***Search***
    - Remove non-alphanumeric characters from search string
    - Read service-id, service-tool-number, service-start-date, service-end-date, service-cost, service-clerk-id from ServiceOrder
    - Search Tool for tool-number equal to service-tool-number
    - Read tool-number, short-desc (aggregate) from Tool
    - Search User for user-id equal to service-clerk-id
    - Perform case insensitive search on tool-number, short-desc (aggregate), service-start-date, service-end-date, service-cost, tool-type and username for search string

```
SELECT * FROM (SELECT so.serviceOrderID, '$' + CAST(CAST(so.service_cost as DECIMAL(10,2)) as VARCHAR(16)) as service_cost, DATE_FORMAT(so.service_start_date, '%Y-%m-%d %H:%i:%s') as service_start_date, DATE_FORMAT(so.service_end_date, '%Y-%m-%d %H:%i:%s') as service_end_date, so.tool_number, u.username, t.short_desc FROM `ServiceOrder` so INNER JOIN `User` u ON so.clerk_email = u.email INNER JOIN `Tool` t ON so.tool_number = t.tool_number WHERE so.service_start_date > NOW() AND so.service_end_date < NOW()) ex WHERE ex.tool_number LIKE '%$SearchTerm%' OR ex.short-desc LIKE '%$SearchTerm%' OR ex.service_start_date LIKE '%$SearchTerm%' OR ex.service_end_date LIKE '%$SearchTerm%' OR ex.service_cost LIKE '%$SearchTerm%' OR ex.username LIKE '%$SearchTerm%';
```

- Upon click ***Fix-Now***
    - Update ServiceOrder service-end-date to ‘now()’ using the record service-id

```
UPDATE `ServiceOrder` set service_end_date = NOW() WHERE tool_number =
$ToolNumber;
```

Sell Tool

NOTE: Aliases are used in this section to keep the queries short.

Abstract Code

- Search Tool for all records not currently “for-sale”
- Read tool-number, short-desc (aggregate), original-price from Tool
- Calculate rental-price to 15% of original-price and deposit-price to 40% of original-price
- Search SaleOrder for records with sale-tool-number equal to tool-number. Limit records from Tool to those without a sale-date.
- Search ServiceOrder for records with service-tool-number equal to tool-number and service-start-date less than today and service-end-date in the future. Limit records from Tool having no active record in ServiceOrder.
- Search Reservation for records with renting-tool-number equal to tool-number, start-date less than ‘now()’, end-date greater than ‘now()’ or pick-up-date less than ‘now()’ and drop-off-date greater than ‘now()’ or without a value. Limit records from Tool having no active record in Reservation.
- Format rental-price and deposit-price as $X.XX

```
SELECT t.tool_number, t.short_desc, CAST(CAST((0.15 * t.original_price) as DECIMAL(10,2)) as VARCHAR(16)) as rental_price, CAST(CAST((0.4 * t.original_price) as DECIMAL(10,2)) as VARCHAR(16)) as deposit_price from `Tool` t LEFT JOIN `SaleOrder` sao ON t.tool_number = sao.tool_number AND sao.sale_date IS NULL LEFT JOIN `ServiceOrder` seo ON t.tool_number = seo.tool_number AND NOT (seo.service_start_date < NOW() AND seo.service_end_date > NOW()) LEFT JOIN `Renting` ren ON t.tool_number = ren.tool_number LEFT JOIN `Reservation` res ON ren.reservationID = res.reservationID AND res.pick-up-date IS NOT NULL AND res.pick-up-date < NOW() AND res.drop_off_date IS NULL;
```

- Populate ***Power Source [dropdown]*** reading distinct power-source values from Tool

```
SELECT DISTINCT tps.power_source from `Tool` t INNER JOIN `ToolPowerSource` tps ON t.tool_number = tps.tool_number WHERE t.tool_type = '$ToolType' ORDER BY tps.power_source;
```

- Populate ***Sub-Type [dropdown]*** reading distinct sub-type values from Tool

```
SELECT DISTINCT sub_type from `Tool` WHERE tool_type = '$ToolType' ORDER BY sub_type;
```

- Upon click ***Type [radio button]***

    - Search Tool for matching tool-type values
    - Read tool-number, short-desc (aggregate), original-price from Tool
    - Calculate rental-price to 15% of original-price and deposit-price to 40% of original-price
    - Search SaleOrder for records with sale-tool-number equal to tool-number. Limit records from Tool to those without a sale-date.
    - Search ServiceOrder for records with service-tool-number equal to tool-number and service-start-date less than today and service-end-date in the future. Limit records from Tool that do not have an active record in ServiceOrder.
    - Search Reservation for records with renting-tool-number equal to tool-number, start-date less than 'now()', end-date greater than 'now()' or pick-up-date less than 'now()' and drop-off-date greater than 'now()' or without a value. Limit records from Tool having no active record in Reservation.
    - Format rental-price and deposit-price as $X.XX

```
SELECT t.tool_number, t.short_desc, CAST(CAST((0.15 * t.original_price) as DECIMAL(10,2)) as VARCHAR(16)) as rental_price, CAST(CAST((0.4 * t.original_price) as DECIMAL(10,2)) as VARCHAR(16)) as deposit_price from `Tool` t LEFT JOIN `SaleOrder` sao ON t.tool_number = sao.tool_number AND sao.sale_date IS NULL LEFT JOIN `ServiceOrder` seo ON t.tool_number = seo.tool_number AND NOT (seo.service_start_date < NOW() AND seo.service_end_date > NOW()) LEFT JOIN `Renting` ren ON t.tool_number = ren.tool_number LEFT JOIN `Reservation` res ON ren.reservationID = res.reservationID AND res.pick-up-date IS NOT NULL AND res.pick-up-date < NOW() AND res.drop_off_date IS NULL WHERE t.tool_type = '$ToolType';
```

- Populate ***Power Source [dropdown]*** reading distinct power-source values from Tool

```
SELECT DISTINCT tps.power_source from `Tool` t INNER JOIN `ToolPowerSource` tps ON t.tool_number = tps.tool_number WHERE t.tool_type = '$ToolType' ORDER BY tps.power_source;
```

- Populate ***Sub-Type [dropdown]*** reading distinct sub-type values from Tool

```
SELECT DISTINCT sub_type from `Tool` WHERE tool_type = '$ToolType' ORDER BY sub_type;
```

- Upon click ***Search***
    - Remove non-alphanumeric characters from search string
    - Read tool-number, short-desc (aggregate), original-price, power-source, sub-type, tool-type from Tool
    - Calculate rental-price to 15% of original-price and deposit-price to 40% of original-price

    - Search SaleOrder for records with sale-tool-number equal to tool-number. Limit records from Tool to those without a sale-date.
    - Search ServiceOrder for records with service-tool-number equal to tool-number and service-start-date less than today and service-end-date in the future. Limit records from Tool that do not have an active record in ServiceOrder.
    - Search Reservation for records with renting-tool-number equal to tool-number, start-date less than ‘now()’, end-date greater than ‘now()’ or pick-up-date less than ‘now()’ and drop-off-date greater than ‘now()’ or without a value. Limit records from Tool having no active record in Reservation.
    - Perform case insensitive search on tool-number, short-desc (aggregate), rental-price, deposit-price, tool-type, sub-type and power-source for search string
    - Format rental-price and deposit-price as $X.XX

```
SELECT DISTINCT * FROM (SELECT t.tool_number, t.tool_type, t.sub_type, tp.t.short_desc, CAST(CAST((0.15 * t.original_price) as DECIMAL(10,2)) as VARCHAR(16)) as rental_price, CAST(CAST((0.4 * t.original_price) as DECIMAL(10,2)) as VARCHAR(16)) as deposit_price from `Tool` t INNER JOIN `ToolPowerSource` tps ON t.tool_number = tps.tool_number AND tps.power_source LIKE '%$SearchTerm%' LEFT JOIN `SaleOrder` sao ON t.tool_number = sao.tool_number AND sao.sale_date IS NULL LEFT JOIN `ServiceOrder` seo ON t.tool_number = seo.tool_number AND NOT (seo.service_start_date < NOW() AND seo.service_end_date > NOW()) LEFT JOIN `Renting` ren ON t.tool_number = ren.tool_number LEFT JOIN `Reservation` res ON ren.reservationID = res.reservationID AND res.pick-up-date IS NOT NULL AND res.pick-up-date < NOW() AND res.drop_off_date IS NULL) ex WHERE ex.tool_number = $SearchTerm OR ex.short_desc LIKE '%$SearchTerm%' OR ex.rental_price LIKE '%$SearchTerm%' OR ex.deposit_price LIKE '%$SearchTerm%' OR ex.tool_type LIKE '%$SearchTerm%' OR ex.sub_type LIKE '%$SearchTerm%' OR ex.power_source LIKE '%$SearchTerm%';
```

- Upon click ***Power Source [dropdown]***
    - Search Tool for matching power-source values
    - Read tool-number, short-desc (aggregate), original-price from Tool
    - Calculate rental-price to 15% of original-price and deposit-price to 40% of original-price
    - Search SaleOrder for records with sale-tool-number equal to tool-number. Limit records from Tool to those without a sale-date.
    - Search ServiceOrder for records with service-tool-number equal to tool-number and service-start-date less than today and service-end-date in the future. Limit records from Tool that do not have an active record in ServiceOrder.
    - Search Reservation for records with renting-tool-number equal to tool-number, start-date less than ‘now()’, end-date greater than ‘now()’ or pick-up-date less than ‘now()’ and drop-off-date greater than ‘now()’ or without a value. Limit records from Tool having no active record in Reservation.
    - Format rental-price and deposit-price as $X.XX

```
SELECT t.tool_number, t.short_desc, CAST(CAST((0.15 * t.original_price) as DECIMAL(10,2)) as VARCHAR(16)) as rental_price, CAST(CAST((0.4 * t.original_price) as DECIMAL(10,2)) as VARCHAR(16)) as deposit_price from `Tool` t INNER JOIN `ToolPowerSource` tps ON t.tool_number = tps.tool_number AND tps.power_source = '$PowerSource' LEFT JOIN `SaleOrder` sao ON t.tool_number = sao.tool_number AND sao.sale_date IS NULL LEFT JOIN `ServiceOrder` seo ON t.tool_number = seo.tool_number AND NOT (seo.service_start_date < NOW() AND seo.service_end_date > NOW()) LEFT JOIN `Renting` ren ON t.tool_number = ren.tool_number LEFT JOIN `Reservation` res ON ren.reservationID = res.reservationID AND res.pick-up-date IS NOT NULL AND res.pick-up-date < NOW() AND res.drop_off_date IS NULL;
```

- Upon click ***Sub-Type [dropdown]***
    - Search Tool for matching sub-type values
    - Read tool-number, short-desc (aggregate), original-price from Tool
    - Calculate rental-price to 15% of original-price and deposit-price to 40% of original-price
    - Search SaleOrder for records with sale-tool-number equal to tool-number. Limit records from Tool to those without a sale-date.
    - Search ServiceOrder for records with service-tool-number equal to tool-number and service-start-date less than today and service-end-date in the future. Limit records from Tool that do not have an active record in ServiceOrder.
    - Format rental-price and deposit-price as $X.XX

```
SELECT t.tool_number, t.short_desc, CAST(CAST((0.15 * t.original_price) as DECIMAL(10,2)) as VARCHAR(16)) as rental_price, CAST(CAST((0.4 * t.original_price) as DECIMAL(10,2)) as VARCHAR(16)) as deposit_price from `Tool` t LEFT JOIN `SaleOrder` sao ON t.tool_number = sao.tool_number AND sao.sale_date IS NULL LEFT JOIN `ServiceOrder` seo ON t.tool_number = seo.tool_number AND NOT (seo.service_start_date < NOW() AND seo.service_end_date > NOW()) LEFT JOIN `Renting` ren ON t.tool_number = ren.tool_number LEFT JOIN `Reservation` res ON ren.reservationID = res.reservationID AND res.pick-up-date IS NOT NULL AND res.pick-up-date < NOW() AND res.drop_off_date IS NULL WHERE t.sub_type = '$SubType';
```

- Upon click ***Sell Tool [button]***
    - Create a new record in SaleOrder
        - Create a new, unique sale-order-id
        - Set sale-tool-number to the selected tool-number
        - Set for-sale-date to ‘now()’
        - Set sale-price to 50% of original-price
        - Set sale-clerk-id to current session user-id

```
INSERT INTO `SaleOrder` (tool_number, sold_by_email, for_sale_date, sale_price) SELECT ('$ToolNumber', '$UserEmail', NOW(), 0.5 * t.original_price) FROM `Tool` t WHERE t.tool_number = $ToolNumber;
```

- Update Tool record as “for-sale”

```
UPDATE `Tool` SET for-sale = TRUE WHERE tool_number = $ToolNumber;
```

View Sale Status

NOTE: Aliases are used in this section to keep the queries short.

Abstract Code

- Search SaleOrder for active records and return sale-id, sale-tool-number, sale-price, for-sale-date, sale-date, sale-clerk-id and sale-customer-id. Format dates as datetime and sale-price as $X.XX.
- Using sale-tool-number from SaleOrder, read Tool for sub-type and short-desc (aggregate), filtering results that match the selected sub-type in ***Type [radio button]***.
- Using sale-customer-id from SaleOrder, read username from User for the matching user record.

```
SELECT so.saleOrderID, so.tool_number, '$' + CAST(CAST(so.sale_price DECIMAL(10,2)) as VARCHAR(16)) as sale_price, DATE_FORMAT(so.for_sale_date, '%Y-%m-%d %H:%i:%s'), DATE_FORMAT(so.sale_date, '%Y-%m-%d %H:%i:%s'), cl.userID as clerkID, cu.username as customer_username, t.short_desc, CASE WHEN so.sale_date IS NOT NULL THEN 'Sold' ELSE 'For-Sale' END as status FROM `SaleOrder` so INNER JOIN `User` cl ON cl.email = so.sold_by_email INNER JOIN `User` cu ON cu.email = so.purchase_by_email INNER JOIN `Tool` t ON t.tool_number = so.tool_number;
```

- If the ***Custom Search [input field]*** is not empty, search sale-id, sale-tool-number, short-desc (aggregate), for-sale-date, sale-date, sale-price and username for matching values.

```
SELECT * FROM (SELECT so.saleOrderID, so.tool_number, '$' + CAST(CAST(so.sale_price DECIMAL(10,2)) as VARCHAR(16)) as sale_price, DATE_FORMAT(so.for_sale_date, '%Y-%m-%d %H:%i:%s') as for_sale_date, DATE_FORMAT(so.sale_date, '%Y-%m-%d %H:%i:%s') as sale_date, cl.userID as clerkID, cu.username as customer_username, t.short_desc, CASE WHEN so.sale_date IS NOT NULL THEN 'Sold' ELSE 'For-Sale' END as status FROM `SaleOrder` so INNER JOIN `User` cl ON cl.email = so.sold_by_email INNER JOIN `User` cu ON cu.email = so.purchase_by_email INNER JOIN `Tool` t ON t.tool_number = so.tool_number) ex WHERE ex.saleOrderID = $SearchTerm OR ex.tool_number = $SearchTerm OR ex.short_desc LIKE '%$SearchTerm%' OR ex.sale_price LIKE '%$SearchTerm%' OR ex.for_sale_date LIKE '%$SearchTerm%' OR ex.sale_date LIKE '%$SearchTerm%' OR ex.customer_username LIKE '%$SearchTerm%';
```

- If a ***Column Header [label]*** has been selected, order the results by the corresponding field for the selected column header, either ascending or descending

```
SELECT * FROM (SELECT so.saleOrderID, so.tool_number, '$' + CAST(CAST(so.sale_price DECIMAL(10,2)) as VARCHAR(16)) as sale_price, DATE_FORMAT(so.for_sale_date,
```

'%Y-%m-%d %H:%i:%s'), DATE_FORMAT(so.sale_date, '%Y-%m-%d %H:%i:%s'), cl.userID as clerkID, cu.username as customer_username, t.short_desc, CASE WHEN so.sale_date IS NOT NULL THEN 'Sold' ELSE 'For-Sale' END as status FROM `SaleOrder` so INNER JOIN `User` cl ON cl.email = so.sold_by_email INNER JOIN `User` cu ON cu.email = so.purchase_by_email INNER JOIN `Tool` t ON t.tool_number = so.tool_number) ORDER BY $HeaderLabel;

- Upon click ***Type [radio button]***
    - Search Tool for matching tool-type values
    - Read tool-number, short-desc (aggregate) from Tool
    - Search SaleOrder for sale-tool-number equal to tool-number
    - Read sale-id, for-sale-date, sale-date, sale-price, sale-clerk-id and sale-customer-id from SaleOrder
    - Format sale-date and for-sale-date as datetime and sale-price as $X.XX.
    - Search User for user-id equal to sale-customer-id
    - Read username from User

SELECT so.saleOrderID, so.tool_number, '$' + CAST(CAST(so.sale_price DECIMAL(10,2)) as VARCHAR(16)) as sale_price, DATE_FORMAT(so.for_sale_date, '%Y-%m-%d %H:%i:%s'), DATE_FORMAT(so.sale_date, '%Y-%m-%d %H:%i:%s'), cl.userID as clerkID, cu.username as customer_username, t.short_desc, CASE WHEN so.sale_date IS NOT NULL THEN 'Sold' ELSE 'For-Sale' END as status FROM `SaleOrder` so INNER JOIN `User` cl ON cl.email = so.sold_by_email INNER JOIN `User` cu ON cu.email = so.purchase_by_email INNER JOIN `Tool` t ON t.tool_number = so.tool_number WHERE t.tool_type = '$ToolType';

- Upon click ***Search***
    - Remove non-alphanumeric characters from search string
    - Read sale-id, sale-tool-number, for-sale-date, sale-date, sale-price, sale-customer-id from SaleOrder
    - Search Tool for tool-number equal to sale-tool-number
    - Read tool-number, short-desc (aggregate) from Tool
    - Search User for user-id equal to sale-customer-id
    - Perform case insensitive search on sale-id, sale-tool-number, sale-clerk-id, short-desc (aggregate), for-sale-date, sale-date, sale-price, tool-type and username for search string

SELECT * FROM (SELECT so.saleOrderID, so.tool_number, '$' + CAST(CAST(so.sale_price DECIMAL(10,2)) as VARCHAR(16)) as sale_price, DATE_FORMAT(so.for_sale_date, '%Y-%m-%d %H:%i:%s') as for_sale_date, DATE_FORMAT(so.sale_date, '%Y-%m-%d %H:%i:%s') as sale_date, cl.userID as clerkID, cu.username as customer_username, t.short_desc, CASE WHEN so.sale_date IS NOT NULL THEN 'Sold' ELSE 'For-Sale' END as status FROM `SaleOrder` so INNER JOIN `User` cl ON cl.email = so.sold_by_email INNER

```
JOIN `User` cu ON cu.email = so.purchase_by_email INNER JOIN `Tool` t ON t.tool_number = so.tool_number) ex WHERE ex.saleOrderID = $SearchTerm OR ex.tool_number = $SearchTerm OR ex.short_desc LIKE '%$SearchTerm%' OR ex.sale_price LIKE '%$SearchTerm%' OR ex.for_sale_date LIKE '%$SearchTerm%' OR ex.sale_date LIKE '%$SearchTerm%' OR ex.customer_username LIKE '%$SearchTerm%';
```

Generate Clerk Report

NOTE: Aliases are used in this section to keep the queries short.

Abstract Code

- Select user-id, first-name, middle-name, last-name, email, hire-date from User / Clerk

```
SELECT u.first_name, u.middle_name, u.last_name u.email, c.hire_date FROM `User` u INNER JOIN `Clerk` c ON u.email = c.email;
```

- Search Reservation for pick-up-email, drop-off-email equal to user-email
- Read pick-up-email, drop-off-email from Reservation
- For each user-email, count number of pick-up-email equal to user-email into number-of-pickups, count number of drop-off-email equal to user-email into number-of-dropoffs
- Calculate combined-total by adding number-of-pickups and number-of-dropoffs
- Format hire-date as datetime
- Sort results by combined-total descending

```
SELECT ex.email, ex.number_of_pickups, ex.number_of_dropoffs, (ex.number_of_pickups + ex.number_of_dropoffs) as combined_total FROM (SELECT IFNULL(pick_up_email, drop_off_email) as email, COUNT(pick_up_email) as number_of_pickups, COUNT(drop_off_email) as number_of_dropoffs FROM `Reservation` res GROUP BY pick_up_email, drop_off_email) ex ORDER BY combined_total DESC;
```

- Upon click ***Back To Report Menu***
    - Leave Clerk Report form
- Upon click ***Reload Results***
    - Select user-email, first-name, middle-name, last-name, email, hire-date from User / Clerk

```
SELECT u.first_name, u.middle_name, u.last_name u.email, c.hire_date FROM `User` u INNER JOIN `Clerk` c ON u.email = c.email;
```

- Search Reservation for pick-up-email, drop-off-email equal to user-email
- Read pick-up-email, drop-off-id from Reservation
- For each user-id, count number of pick-up-email equal to user-email into number-of-pickups, count number of drop-off-email equal to user-email into number-of-dropoffs
- Calculate combined-total by adding number-of-pickups and number-of-dropoffs
- Format hire-date as datetime
- Sort results by combined-total descending

```
SELECT ex.email, ex.number_of_pickups, ex.number_of_dropoffs, (ex.number_of_pickups + ex.number_of_dropoffs) as combined_total FROM (SELECT IFNULL(pick_up_email, drop_off_email) as email, COUNT(pick_up_email) as number_of_pickups, COUNT(drop_off_email) as number_of_dropoffs FROM `Reservation` res GROUP BY pick_up_email, drop_off_email) ex ORDER BY combined_total DESC;
```

Generate Customer Report

NOTE: Aliases are used in this section to keep the queries short.

Abstract Code

- Read user-id, first-name, middle-name, last-name, email, phone (where primary is true) from User

```
SELECT u.first_name, u.middle_name, u.last_name, u.email, (CAST(cp.area-code AS VARCHAR(3)) + CAST(cp.phone-number AS VARCHAR(7))) as phone FROM `User` u INNER JOIN `Customer` c ON u.email = c.email LEFT JOIN `CustomerPhoneNumber` cp ON (c.email = cp.email AND cp.primary = TRUE);
```

- Search Reservation for renting-customer-id equal to user-id. Limit records from User by matching records with ‘now()’ - pick-up-date < 30 days.
- Read tool-number, start-date, pick-up-date from Reservation
- For each user-id, count number of renting-customer-id where start-date is not null into number-of-reservations, count number of tool-number where pick-up-date is not null equal into number-of-tools-rented
- Sort results by number-of-tools-rented descending

```
SELECT ren.email, SUM(CASE WHEN res.start_date IS NOT NULL THEN 1 ELSE 0 END) as number_of_reservations, SUM(CASE WHEN res.pick_up_date IS NOT NULL THEN 1
```

```
ELSE 0 END) as number_of_tools_rented FROM `Renting` ren INNER JOIN `Reservation` res ON ren.reservationID = res.reservationID WHERE DATEDIFF(now(), pick_up_date) < 30 GROUP BY ren.email ORDER BY number_of_tools_rented DESC;
```

- Upon click ***Back To Report Menu***
    - Leave Customer Report form
- Upon click ***Reload Results***
    - Read user-id, first-name, middle-name, last-name, email, phone (where primary is true) from User

```
SELECT u.first_name, u.middle_name, u.last_name, u.email, (CAST(cp.area-code AS VARCHAR(3)) + CAST(cp.phone-number AS VARCHAR(7))) as phone FROM `User` u INNER JOIN `Customer` c ON u.email = c.email LEFT JOIN `CustomerPhoneNumber` cp ON (c.email = cp.email AND cp.primary = TRUE);
```

    - Search Reservation for renting-customer-id equal to user-id. Limit records from User by matching records with ‘now()’ - pick-up-date < 30 days.
    - Read tool-number, start-date, pick-up-date from Reservation
    - For each user-id, count number of renting-customer-id where start-date is not null into number-of-reservations, count number of tool-number where pick-up-date is not null equal into number-of-tools-rented
    - Sort results by number-of-tools-rented descending

```
SELECT ren.email, SUM(CASE WHEN res.start_date IS NOT NULL THEN 1 ELSE 0 END) as number_of_reservations, SUM(CASE WHEN res.pick_up_date IS NOT NULL THEN 1 ELSE 0 END) as number_of_tools_rented FROM `Renting` ren INNER JOIN `Reservation` res ON ren.reservationID = res.reservationID WHERE DATEDIFF(now(), pick_up_date) < 30 GROUP BY ren.email ORDER BY number_of_tools_rented DESC;
```

- Upon click **View Profile**
    - Navigate to View Customer Profile

Generate Tool Report

NOTE: Aliases are used in this section to keep the queries short.

Abstract Code

- Read tool-number, short-desc, original-price from Tool

```
SELECT tool_number, short_desc, original_price FROM `Tool`;
```

- Select from Reservation where renting tool_number equals tool-number
- Read tool-number, start-date, pick-up-date, end-date, drop-off-date from Reservation

```
SELECT t.tool_number, t.short_desc, t.original_price, res.start_date, res.end_date,
res.pick_up_date, res.drop_off_date FROM `Tool` t INNER JOIN `Renting` ren ON
ren.tool_number = t.tool_number INNER JOIN `Reservation` res ON ren.reservationID =
res.reservationID;
```

- Select from SaleOrder where sale-tool-number equals tool-number
- Read for-sale-date, sale-date, sale-price from SaleOrder

```
SELECT t.tool_number, t.short_desc, t.original_price, so.for_sale_date, so.sale_date,
so.sale_price FROM `Tool` t INNER JOIN `SaleOrder` so ON so.tool_number =
t.tool_number;
```

- Select from ServiceOrder where service-tool-number equals tool-number
- Read service-start-date, service-end-date, service-cost from ServiceOrder

```
SELECT t.tool_number, t.short_desc, t.original_price, so.service_start_date,
so.service_end_date, so.service_cost FROM `Tool` t INNER JOIN `ServiceOrder` so ON
so.tool_number = t.tool_number;
```

- For each tool-number, calculate the number of days rented for each matching reservation record as ceiling(drop-off-date (or ‘now()’ if null) - pick-up-date) * tool-rental-price into rental-profit.

```
SELECT t.tool_number, CAST(SUM(DATEDIFF(res.drop_off_date,
IFNULL(res.pick_up_date, NOW())) * 0.15 * t.original_price) as DECIMAL(10, 2)) as
rental_profit FROM `Tool` t INNER JOIN `Renting` ren ON t.tool_number = ren.tool_number
INNER JOIN `Reservation` res ON ren.reservationID = res.reservationID GROUP BY
t.tool_number, t.original_price;
```

- For each tool-number, calculate the value of total-cost by taking the difference of the original-price and the sum of all matching service order records.

```
SELECT t.tool_number, CAST((t.original_price - SUM(so.service_cost)) as DECIMAL(10,2)) as total_cost FROM `Tool` t INNER JOIN `ServiceOrder` so ON t.tool_number = so.tool_number GROUP BY t.tool_number, t.original_price;
```

- For each tool-number, calculate the total-profit as the difference between rental-profit and total-cost.
- For each tool-number, calculate the current-status as
    - If matching sale order has sale-order-id not null and sale-date not null, set to “Sold”
        - Set status-date to sale-date
    - If tool has for-sale set to True, set to “For-Sale”
        - Set status-date to for-sale-date
    - If matching service order has service-start-date < ‘now()’ and service-end-date > ‘now()’ or null, set to “In-Repair”
        - Set status-date to service-start-date
    - If matching reservation has start-date < ‘now()’ or pick-up-date < ‘now()’ and end-date > ‘now()’ and drop-off-date is null, set to “Rented”
        - Set status-date to pick-up-date if pick-up-date is not equal to start-date, else start-date
    - If not any other condition, set to “Available”
        - Set status-date to null
- Sort results by total-profit descending
- Format rental-profit, total-cost and total-profit as $X.XX
- Format dates as date (mm/dd/yyyy)

```
SELECT t.tool_number, t.short_desc, tot_rent.rental_profit, tot_cost.total_cost, CAST((tot_rent.rental_profit - tot_cost.total_cost) AS DECIMAL(10, 2)) as total_profit, CASE WHEN sao.sale_date IS NOT NULL AND sao.sale_date < NOW() THEN sao.sale_date ELSE WHEN sao.for_sale_date IS NOT NULL AND sao.for_sale_date < NOW() THEN sao.for_sale_date ELSE WHEN seo.service_start_date IS NOT NULL AND seo.service_start_date < NOW() AND (seo.service_end_date IS NULL OR seo.service_end_date > NOW()) THEN seo.service_start_date ELSE WHEN res_date.pick_up_date IS NOT NULL THEN res_date.pick_up_date ELSE NULL END as date, CASE WHEN sao.sale_date IS NOT NULL AND sao.sale_date < NOW() THEN 'Sold' ELSE WHEN sao.for_sale_date IS NOT NULL AND sao.for_sale_date < NOW() THEN 'For-Sale' ELSE WHEN seo.service_start_date IS NOT NULL AND seo.service_start_date < NOW() AND (seo.service_end_date IS NULL OR seo.service_end_date > NOW()) THEN 'In-Repair' ELSE WHEN res_date.pick_up_date IS NOT NULL THEN 'Rented' ELSE 'Available' END as status FROM `Tool` t LEFT JOIN
```

```
`SaleOrder` sao ON t.tool_number = sao.tool_number LEFT JOIN `ServiceOrder` seo ON
t.tool_number = seo.tool_number LEFT JOIN (Select ren.tool_number, res.pick_up_date
FROM `Renting` ren LEFT JOIN `Reservation` res ON ren.reservation_id = res.reservation_id
WHERE res.pick_up_date < NOW() AND res.drop_off_date IS NULL) AS res_date ON
t.tool_number = res_date.tool_number LEFT JOIN (SELECT t.tool_number,
CAST(SUM(DATEDIFF(res.drop_off_date, IFNULL(res.pick_up_date, NOW())) * 0.15 *
t.original_price) as DECIMAL(10, 2)) as rental_profit FROM `Tool` t INNER JOIN `Renting`
ren ON t.tool_number = ren.tool_number INNER JOIN `Reservation` res ON
ren.reservationID = res.reservationID GROUP BY t.tool_number, t.original_price) tot_rent ON
t.tool_number = tot_rent.tool_number LEFT JOIN (SELECT t.tool_number,
CAST((t.original_price - SUM(so.service_cost)) as DECIMAL(10,2)) as total_cost FROM
`Tool` t INNER JOIN `ServiceOrder` so ON t.tool_number = so.tool_number WHERE
t.tool_type = '$ToolType' GROUP BY t.tool_number, t.original_price) tot_cost ON
t.tool_number = tot_cost.tool_number;
```

- Upon click ***Type [radio button]***
    - Read tool-number, short-desc (aggregate), original-price from Tool where tool-type matches selected ***Type [radio button]***

```
SELECT tool_number, short_desc, original_price FROM `Tool` WHERE t.tool_type =
'$ToolType';
```

    - Select from Reservation where renting-tool-number equals tool-number
    - Read tool-number, tool-rental-price, start-date, pick-up-date, end-date, drop-off-date from Reservation

```
SELECT t.tool_number, t.short_desc, t.original_price, res.start_date, res.end_date,
res.pick_up_date, res.drop_off_date FROM `Tool` t INNER JOIN `Renting` ren ON
ren.tool_number = t.tool_number INNER JOIN `Reservation` res ON ren.reservationID =
res.reservationID WHERE t.tool_type = '$ToolType';
```

    - Select from SaleOrder where sale-tool-number equals tool-number
    - Read for-sale-date, sale-date, sale-price from SaleOrder

```
SELECT t.tool_number, t.short_desc, t.original_price, so.for_sale_date, so.sale_date,
so.sale_price FROM `Tool` t INNER JOIN `SaleOrder` so ON so.tool_number = t.tool_number
WHERE t.tool_type = '$ToolType';
```

    - Select from ServiceOrder where service-tool-number equals tool-number
    - Read service-start-date, service-end-date, service-cost from ServiceOrder

```
SELECT t.tool_number, t.short_desc, t.original_price, so.service_start_date,
so.service_end_date, so.service_cost FROM `Tool` t INNER JOIN `ServiceOrder` so ON
so.tool_number = t.tool_number WHERE t.tool_type = '$ToolType';
```

- For each tool-number, calculate the number of days rented for each matching reservation record as ceiling(drop-off-date (or 'now()' if null) - pick-up-date) * tool-rental-price into rental-profit.

```
SELECT t.tool_number, CAST(SUM(DATEDIFF(res.drop_off_date,
IFNULL(res.pick_up_date, NOW())) * 0.15 * t.original_price) as DECIMAL(10, 2)) as
rental_profit FROM `Tool` t INNER JOIN `Renting` ren ON t.tool_number = ren.tool_number
INNER JOIN `Reservation` res ON ren.reservationID = res.reservationID WHERE t.tool_type
= '$ToolType' GROUP BY t.tool_number, t.original_price;
```

- For each tool-number, calculate the value of total-cost by taking the difference of the original-price and the sum of all matching service order records.

```
SELECT t.tool_number, CAST((t.original_price - SUM(so.service_cost)) as DECIMAL(10,2))
as total_cost FROM `Tool` t INNER JOIN `ServiceOrder` so ON t.tool_number =
so.tool_number WHERE t.tool_type = '$ToolType' GROUP BY t.tool_number, t.original_price;
```

- For each tool-number, calculate the total-profit as the difference between rental-profit and total-cost.
- For each tool-number, calculate the current-status as
  - If matching sale order has sale-order-id not null and sale-date not null, set to "Sold"
    - Set status-date to sale-date
  - If tool has for-sale set to True, set to "For-Sale"
    - Set status-date to for-sale-date
  - If matching service order has service-start-date < 'now()' and service-end-date > 'now()' or null, set to "In-Repair"
    - Set status-date to service-start-date
  - If matching reservation has start-date < 'now()' or pick-up-date < 'now()' and end-date > 'now()' and drop-off-date is null, set to "Rented"
    - Set status-date to pick-up-date if pick-up-date is not equal to start-date, else start-date
  - If not any other condition, set to "Available"
    - Set status-date to null
- Sort results by total-profit descending
- Format rental-profit, total-cost and total-profit as $X.XX
- Format dates as date (mm/dd/yyyy)

```
SELECT t.tool_number, t.short_desc, tot_rent.rental_profit, tot_cost.total_cost,
CAST((tot_rent.rental_profit - tot_cost.total_cost) AS DECIMAL(10, 2)) as total_profit, CASE
WHEN sao.sale_date IS NOT NULL AND sao.sale_date < NOW() THEN sao.sale_date ELSE
WHEN sao.for_sale_date IS NOT NULL AND sao.for_sale_date < NOW() THEN
sao.for_sale_date ELSE WHEN seo.service_start_date IS NOT NULL AND
seo.service_start_date < NOW() AND (seo.service_end_date IS NULL OR
seo.service_end_date > NOW()) THEN seo.service_start_date ELSE WHEN
res_date.pick_up_date IS NOT NULL THEN res_date.pick_up_date ELSE
NULL END as date, CASE WHEN sao.sale_date IS NOT NULL AND sao.sale_date < NOW()
THEN 'Sold' ELSE WHEN sao.for_sale_date IS NOT NULL AND sao.for_sale_date < NOW()
THEN 'For-Sale' ELSE WHEN seo.service_start_date IS NOT NULL AND
seo.service_start_date < NOW() AND (seo.service_end_date IS NULL OR
seo.service_end_date > NOW()) THEN 'In-Repair' ELSE WHEN res_date.pick_up_date IS
NOT NULL THEN 'Rented' ELSE 'Available' END as status FROM `Tool` t LEFT JOIN
`SaleOrder` sao ON t.tool_number = sao.tool_number LEFT JOIN `ServiceOrder` seo ON
t.tool_number = seo.tool_number LEFT JOIN (Select ren.tool_number, res.pick_up_date
FROM `Renting` ren LEFT JOIN `Reservation` res ON ren.reservation_id = res.reservation_id
WHERE res.pick_up_date < NOW() AND res.drop_off_date IS NULL) AS res_date ON
t.tool_number = res_date.tool_number LEFT JOIN (SELECT t.tool_number,
CAST(SUM(DATEDIFF(res.drop_off_date, IFNULL(res.pick_up_date, NOW())) * 0.15 *
t.original_price) as DECIMAL(10, 2)) as rental_profit FROM `Tool` t INNER JOIN `Renting`
ren ON t.tool_number = ren.tool_number INNER JOIN `Reservation` res ON
ren.reservationID = res.reservationID GROUP BY t.tool_number, t.original_price) tot_rent ON
t.tool_number = tot_rent.tool_number LEFT JOIN (SELECT t.tool_number,
CAST((t.original_price - SUM(so.service_cost)) as DECIMAL(10,2)) as total_cost FROM
`Tool` t INNER JOIN `ServiceOrder` so ON t.tool_number = so.tool_number WHERE
t.tool_type = '$ToolType' GROUP BY t.tool_number, t.original_price) tot_cost ON
t.tool_number = tot_cost.tool_number WHERE t.tool_type = '$ToolType';
```

- Upon click ***Search***
    - Remove non-alphanumeric characters from search string
    - Read tool-number, short-desc (aggregate), original-price from Tool where tool-type matches selected ***Type [radio button]***

```
SELECT tool_number, short_desc, original_price FROM `Tool` WHERE t.tool_type =
'$ToolType';
```

    - Select from Reservation where renting-tool-number equals tool-number
    - Read tool-number, tool-rental-price, start-date, pick-up-date, end-date, drop-off-date from Reservation

```
SELECT t.tool_number, t.short_desc, t.original_price, res.start_date, res.end_date,
res.pick_up_date, res.drop_off_date FROM `Tool` t INNER JOIN `Renting` ren ON
ren.tool_number = t.tool_number INNER JOIN `Reservation` res ON ren.reservationID =
res.reservationID WHERE t.tool_type = '$ToolType';
```

- Select from SaleOrder where sale-tool-number equals tool-number
- Read for-sale-date, sale-date, sale-price from SaleOrder

```
SELECT t.tool_number, t.short_desc, t.original_price, so.for_sale_date, so.sale_date,
so.sale_price FROM `Tool` t INNER JOIN `SaleOrder` so ON so.tool_number = t.tool_number
WHERE t.tool_type = '$ToolType';
```

- Select from ServiceOrder where service-tool-number equals tool-number
- Read service-start-date, service-end-date, service-cost from ServiceOrder

```
SELECT t.tool_number, CAST((t.original_price - SUM(so.service_cost)) as DECIMAL(10,2))
as total_cost FROM `Tool` t INNER JOIN `ServiceOrder` so ON t.tool_number =
so.tool_number WHERE t.tool_type = '$ToolType' GROUP BY t.tool_number, t.original_price;
```

- For each tool-number, calculate the number of days rented for each matching reservation record as ceiling(drop-off-date (or ‘now()’ if null) - pick-up-date) * tool-rental-price into rental-profit.

```
SELECT t.tool_number, CAST(SUM(DATEDIFF(res.drop_off_date,
IFNULL(res.pick_up_date, NOW())) * 0.15 * t.original_price) as DECIMAL(10, 2)) as
rental_profit FROM `Tool` t INNER JOIN `Renting` ren ON t.tool_number = ren.tool_number
INNER JOIN `Reservation` res ON ren.reservationID = res.reservationID WHERE t.tool_type
= '$ToolType' GROUP BY t.tool_number, t.original_price;
```

NOTE: rental_profit is left as a decimal becuase it will be used to calculate the rental_profit

- For each tool-number, calculate the value of total-cost by taking the difference of the original-price and the sum of all matching service order records.

```
SELECT t.tool_number, CAST((t.original_price - SUM(so.service_cost)) as DECIMAL(10,2))
as total_cost FROM `Tool` t INNER JOIN `ServiceOrder` so ON t.tool_number =
so.tool_number WHERE t.tool_type = '$ToolType' GROUP BY t.tool_number, t.original_price;
```

NOTE: total_cost is left as decimal because it will be used to calculate the total_profit.

- For each tool-number, calculate the total-profit as the difference between rental-profit and total-cost.
- For each tool-number, calculate the current-status as
  - If matching sale order has sale-order-id not null and sale-date not null, set to “Sold”
    - Set status-date to sale-date
  - If tool has for-sale set to True, set to “For-Sale”
    - Set status-date to for-sale-date
  - If matching service order has service-start-date < ‘now()’ and service-end-date > ‘now()’ or null, set to “In-Repair”
    - Set status-date to service-start-date
  - If matching reservation has start-date < ‘now()’ or pick-up-date < ‘now()’ and end-date > ‘now()’ and drop-off-date is null, set to “Rented”
    - Set status-date to pick-up-date if pick-up-date is not equal to start-date, else start-date
  - If not any other condition, set to “Available”
    - Set status-date to null
- Perform case insensitive search on tool-number, current-status, status-date, short-desc, rental-profit, total-cost, total-profit for search string
- Sort results by total-profit descending
- Format rental-profit, total-cost and total-profit as $X.XX
- Format dates as date (mm/dd/yyyy)

```
SELECT t.tool_number, t.short_desc, tot_rent.rental_profit, tot_cost.total_cost,
CAST((tot_rent.rental_profit - tot_cost.total_cost) AS DECIMAL(10, 2)) as total_profit, CASE
WHEN sao.sale_date IS NOT NULL AND sao.sale_date < NOW() THEN sao.sale_date ELSE
WHEN sao.for_sale_date IS NOT NULL AND sao.for_sale_date < NOW() THEN
sao.for_sale_date ELSE WHEN seo.service_start_date IS NOT NULL AND
seo.service_start_date < NOW() AND (seo.service_end_date IS NULL OR
seo.service_end_date > NOW()) THEN seo.service_start_date ELSE WHEN
res_date.pick_up_date IS NOT NULL THEN res_date.pick_up_date ELSE
NULL END as date, CASE WHEN sao.sale_date IS NOT NULL AND sao.sale_date < NOW()
THEN 'Sold' ELSE WHEN sao.for_sale_date IS NOT NULL AND sao.for_sale_date < NOW()
THEN 'For-Sale' ELSE WHEN seo.service_start_date IS NOT NULL AND
seo.service_start_date < NOW() AND (seo.service_end_date IS NULL OR
seo.service_end_date > NOW()) THEN 'In-Repair' ELSE WHEN res_date.pick_up_date IS
NOT NULL THEN 'Rented' ELSE 'Available' END as status FROM `Tool` t LEFT JOIN
`SaleOrder` sao ON t.tool_number = sao.tool_number LEFT JOIN `ServiceOrder` seo ON
t.tool_number = seo.tool_number LEFT JOIN (Select ren.tool_number, res.pick_up_date
FROM `Renting` ren LEFT JOIN `Reservation` res ON ren.reservation_id = res.reservation_id
WHERE res.pick_up_date < NOW() AND res.drop_off_date IS NULL) AS res_date ON
t.tool_number = res_date.tool_number LEFT JOIN (SELECT t.tool_number,
CAST(SUM(DATEDIFF(res.drop_off_date, IFNULL(res.pick_up_date, NOW())) * 0.15 *
t.original_price) as DECIMAL(10, 2)) as rental_profit FROM `Tool` t INNER JOIN `Renting`
```

```
ren ON t.tool_number = ren.tool_number INNER JOIN `Reservation` res ON
ren.reservationID = res.reservationID GROUP BY t.tool_number, t.original_price) tot_rent ON
t.tool_number = tot_rent.tool_number LEFT JOIN (SELECT t.tool_number,
CAST((t.original_price - SUM(so.service_cost)) as DECIMAL(10,2)) as total_cost FROM
`Tool` t INNER JOIN `ServiceOrder` so ON t.tool_number = so.tool_number WHERE
t.tool_type = '$ToolType' GROUP BY t.tool_number, t.original_price) tot_cost ON
t.tool_number = tot_cost.tool_number WHERE t.tool_type = '$ToolType';
```

NOTE: Search by search term is not included in this query as it would add another sub-select statement to the SQL. It is excluded for brevity.