Table of Contents:

Login User

Abstract Code

- User enters *Username* ($username), *Password* ($password) input fields.
- If data validation is successful for both username and password input fields, then when ***Login*** button is clicked:
    - System queries the User table for the user's input of username($username) and password ($password):

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

    - If a valid entry is found in User table, then:
        - Store username as session variable '$Username'
        - The system then queries if the user is an administrative user by validating his/her ($username) in the AdministrativeUser table

```
SELECT username FROM AdministrativeUser WHERE username =
'$username';
```

            - If the query result is successful, then:
                - Store user type session variable as '$UserType' with value as 'AdminUser'
            - Else if the query is not found:
                - Store user type session variable as '$UserType' with value as 'RegularUser'
            - Go the **Main Menu** form
    - Else If the user's record is found but User.password ! = '$password'.
        - Go back to the **Login** form, with an error message.
- If *Username* and *Password* inputs are invalid, display **Login** form, with error message.

Register User

Abstract Code

- User enters *FirstName* ($first_name), *LastName* ($last_name), *Username* ($username), *Password* ($password) and *ConfirmPassword*($confirm_password) input fields in **Register** form**.**
- If data validation is successful for $username, $password and $confirm_password values, then when ***Register*** button is clicked,
    - Check an existing record with same username is found in User table:

```
SELECT username from `User` WHERE username = '$username';
```

    - If an existing record with same $username is found
        - Go back to the **Register** form, with the error message.
    - Else
        - System inserts the value of user's input for username($username), first_name($first_name), last_name ($last_name) and password($password) to User table

```
INSERT INTO `User` (username, first_name, last_name, password)
VALUES ('$username', '$first_name', '$last_name', '$password');
```

        - Store username as session variable '$Username'
        - Go to the **Main Menu** form.
    - Else if any input fields are invalid, display **Register** form, with error message.
- If the database administrator would need to consider this user as as administrative user, below query can be used to insert user's username('$username') and position('$position') into AdministartiveUser table

```
INSERT INTO AdministrativeUser (username, position) VALUES ('$username', '$position');
```

Display Main Menu

Abstract Code

- System queries the User table to get the $first_name and $last_name of the current user using the session variable '$UserId'

```
SELECT first_name, last_name from User WHERE username = '$Username';
```

- Display the $first_name and $last_name of the current user
- If the variable '$UserType' belongs to an AdministrativeUser, then:
    - System queries the position from AdministrativeUser table

```
SELECT position from AdministrativeUser WHERE username = '$username';
```

    - Display the position of Admin user
- If the variable '$UserType' value is 'RegularUser', show Auction Options only with below links:
    - "***Search for Items***", "***List Item***" and "***View Auction Results***"
- Else If '$UserType' value is 'AdminUser', show all the above Auction Options and include the links to Reports below:
    - "***Category Report***", "***User Report***", "***Top Rated Items***," "***Auction Statistics***" and "***Canceled Auction Details***"

- Upon clicking:
  - ***Search for Items*** button - Jump to the **Display Search Result** task.
  - ***List Item*** button - Jump to the **List Items for Sale** task.
  - ***View Auction Results*** button - Jump to the **Display Auction Results** task.
  - ***Category Report*** button - Jump to the **Display Category Information** task.
  - ***User Report*** button - Jump to the **Display User Information** task.
  - ***Top Rated Items*** button - Jump to the **Display Top Rated Items** task.
  - ***Auction Statistics*** button - Jump to the **Display Auction Statistics** task.
  - ***Canceled Auction Details*** button - Jump to the **Display Canceled Auction Details** task

List Item for Sale

Abstract Code

- User clicked on the ***List Item*** button from the **Main Menu** form.
  - Query for the list of available categories from the Category table

```
SELECT category_name from Category;
```

  - Display the **New Item for Auction** form:
    - Input fields for *Item Name, Description, start auction bidding at, Minimum sale price, Get it Now price.*
    - Dropdown with query result for *Category*
    - Dropdown for *Condition* with values as New, Very Good, Good, Fair and Poor.
    - Dropdown with values of 1,3,5 and 7 days for *Auction ends in*.
    - Checkbox for *Returns Accepted?*
    - Buttons for ***Cancel*** and ***List My Item***
  - User enters item details in all the input fields and selects value from the dropdown: *Item Name* ($item_name), *Description* ($item_description), *Category* ($category_name), *Start auction bidding at* ($starting_bid), *Minimum sale price* ($min_sale_price), *Auction ends in* ($auction_length), *Get It Now price* ($get_it_now_price), and Return Accepted ($returnable)
  - If data validations are successful for all input fields:
    - When the ***List My Item*** button is clicked, add a new record in the Item table with all details from the input fields.

```
INSERT INTO Item (item_name, item_description, item_condition, returnable,
starting_bid, min_sale_price, auction_end_time, get_it_now_price, listeduser,
category_name, cancelled_username, cancelled_reason)
VALUES ('$item_name', '$item_description', '$category_name', '$item_condition',
$returnable, $starting_bid, $min_sale_price, DATE_ADD(NOW(), INTERVAL
$auction_length DAY),$get_it_now_price,
'$listeduser','$category_name','$cancelled_username','$cancelled_reason');
```

        - Go to the **Main Menu** form
    - When ***Cancel*** button is clicked:
        - Go back to the **Main Menu** form.
- Else if any of the input fields are invalid, display the **List Item for Sale** form, with the error message.

Display Search Result

Abstract Code

- User click o*n **Search for Items*** button from **Main Menu** page
- Run the **Display Search Result** task: query for items matching the search criteria and display the results.
    - Find the list of available categories from the Category table

```
SELECT category_name from Category;
```

    - Display the **Item Search** form with user's input value of:
        - Keyword ($keyword) , Minimum Price($min_price) and Maximum Price($max_price) into input fields
        - Category dropdown with values from the category select query result and a blank value
        - Condition at least dropdown with values including "New", "Very Good", "Good", "Fair", "Poor" and a blank value.
        - **Cancel** and **Search** buttons.
    - If data validation is successful for $keyword, $min_price, $max_price, $category and $condition values to accept either float values or blank, then:
        - When the **Search** button is clicked: build a search query based on the values from the **Item Search** form and run the consolidated query on the Item and Bid table to get the matched list of items.

```
SELECT
        i1.itemID AS ID,
        i1.item_name AS 'Item Name',
        b1.bid_amount AS 'Current Bid',
        b1.username AS 'High Bidder',
        i1.get_it_now_price AS 'Get It Now Price',
        i1.auction_end_time AS 'Auction Ends'
FROM
        Item i1
LEFT JOIN (
        SELECT
                b.itemID,
                b.bid_amount,
                b.username
        FROM Bid b
        INNER JOIN (
                SELECT
                        itemID,
                        MAX(bid_amount) AS MaxBid
                FROM Bid
                GROUP BY itemID
        ) AS MaxBids ON b.itemID = MaxBids.itemID AND b.bid_amount = MaxBids.MaxBid
) AS b1 ON i1.itemID = b1.itemID

WHERE
        i1.itemID IN (
                SELECT
                        i.itemID
                FROM Item i
                LEFT JOIN Bid ON i.itemID = Bid.itemID
                WHERE
                        ('$keyword' IS NULL OR BINARY item_name LIKE CONCAT('%', '$keyword', '%') OR BINARY
                        item_description LIKE CONCAT('%', '$keyword', '%'))
                        AND ('$category' IS NULL OR category_name = '$category')
                        AND ($min_price IS NULL OR (SELECT MAX(bid_amount) FROM Bid WHERE Item.itemID = Bid.itemID)
>= $min_price OR starting_bid >= $min_price)
                        AND ($max_price IS NULL OR (SELECT MAX(bid_amount) FROM Bid WHERE Item.itemID = Bid.itemID)
<= $max_price OR starting_bid <= $max_price)
                        AND ('$condition' IS NULL OR item_condition >= '$condition')
                GROUP BY i.itemID
        );
```

- If the query has output from Item table, then:
    - Display **View Search Results Listing** form that displays the query result
        - ***Back to Search*** button
    - When any one of the item names is clicked:
        - Store the corresponding $itemID as session variable '$ItemId'
        - Store the $item_name as session variable '$ItemName'
        - Go to the **Item for Sale** form.
    - When ***Back to Search*** button is clicked:
        - Go back to **Item Search** form.
- Else if the search query returned no output:
    - Go back to **Item Search** form, with an error message.
- When the Cancel button is clicked, go to the **Main Menu** form.
- Else *Minimum Price* and *Maximum Price* input fields are invalid, display the **Item Search** form, with error message.

Item Bidding

Abstract Code

- User clicked on one of Item Name from **View Search Results Listing** form.
- Run the **Item Bidding** task: query the Item table for information about the item where the session variable '$ItemID' is the ID of the item, display information about the item including the latest four bids and write to Bid & Auction based on the details input from the user.
    - Run the **Item Lookup** subtask to query item details:

```
SELECT itemID, item_name, item_description, category_name, item_condition,
returnable, get_it_now_price, auction_end_time FROM Item WHERE itemID=$ItemID;
```

    - Display the **Item for Sale** form:
        - Display the $itemID, '$item_name', '$item_description', '$category_name', '$item_condition'.
        - If $returnable has value of true, then:
            - Display Returns Accepted with a *checkbox ticked*.
        - Else:
            - Display Returns Accepted with a *checkbox not ticked*.
        - If $get_it_now_price is not NULL, then:
            - Display the GetItNowPrice value
            - Display a ***Get It Now!*** Button
        - Display Auction Ended with the $auction_end_time value
        - Find the latest four bids of current Item that is highest and most recent from Bid:

```
SELECT bid_amount AS `Bid Amount`,time_of_bid AS `Time of Bid`,
username AS `Username`
FROM Bid WHERE itemID=$itemID
ORDER BY bid_amount DESC LIMIT 4;
```

            - Display the query result with bid information.
        - Display *Your Bid* input field
            - Find the highest bid of the current item

```
SELECT MAX(bid_amount) AS 'MaxBid' FROM Bid WHERE itemID=$itemID GROUP BY itemID;
```

            - Calculate the acceptable minimum bid as MaxBid+1

- Display the minimum bid as informational text below the input field.

- Find the user who listed the item from the Item table.

```
SELECT listeduser FROM Item WHERE itemID=$itemID;
```

- If the session variable '$Username' is not equal to '$listeduser':
  - Display ***Bid on this Item*** and ***Close*** buttons.
  - Display link to **View Ratings** form.
- Else if '$Username' is equal to '$listeduser', then:
  - Display links to **View Ratings** and **Edit Description** forms.
  - Display ***Close*** button only.
- Check if current logged in user belongs to an AdministrativeUser using session variable '$Username'

```
SELECT username FROM AdministrativeUser WHERE username='$Username';
```

  - If the query returns a record, then display all the buttons and form links along with:
    - ***Cancel This Item*** button.

- When the ***Get It Now!*** Button is clicked:
  - Run the **Bid Creation** subtask to create a new entry in Bid with GetItNowPrice value as BidAmount

```
INSERT INTO Bid (itemID, username,time_of_bid,bid_amount)
VALUES ($itemID, '$Username', NOW(), $get_it_now_price);
```

  - Update the current system time as AuctionEndTime in the Item table.

```
UPDATE Item SET auction_end_time=NOW() WHERE itemID=$itemID;
```

- When user enters value in *Your Bid* input field and click on ***Bid on this Item*** button:
  - If data validation is successful for *Your Bid* input field to accept a float value greater than or equal to minimum bid, then:
    - Run the **Bid Creation** subtask to create a new entry in Bid

```
INSERT INTO Bid (itemID, username,time_of_bid,bid_amount)
VALUES ($itemID, '$Username', NOW(), $bid_amount);
```

        - Else *Your Bid* input field is invalid, display **Item for Sale** form with error message.
    - When ***Cancel This Item*** button is clicked:
        - Display a popup with one input field to get the $cancelled_reason
        - Run the Cancel Auction subtask to get the mark the item as canceled in Item:

```
UPDATE Item
SET auction_end_time=NOW(),
      cancelled_username='$Username',
      cancelled_reason='$cancelled_reason'
 WHERE itemID=$itemID;
```

    - Upon clicking:
        - **View Ratings** form link – Jump to **View Item Ratings** task.
        - **Edit Description** form link – Jump to **Edit Item Description** task.
        - ***Close*** button – Go back to the **View Search Results Listing** screen.

View Rating

Abstract Code

- User clicked on ***View Ratings*** link from **Item for Sale** form
- Run the **View Ratings** task to query all ratings related to the current item and display:
    - Display the Item ID as the value from session variable '$ItemID'.
    - Display the Item Name as the value from session variable '$ItemName'.
    - Run the query against Rating table to find the average star rating of all ratings with identical name and display the value

```
SELECT
        ROUND(AVG(r.star), 1) AS `Average Rating`
FROM Rating r
INNER JOIN Item i ON r.itemID = i.itemID
WHERE BINARY i.item_name='$ItemName'
```

    - Find all ratings from Rating where item name is same as session variable '$ItemName'.
        - Find the “Rated By” user by calculating winner of each item in Rating

```
SELECT r1.itemID, r2.Winner AS 'Rated By', r1. rating_date_time AS 'Date', r1.star
AS 'Star', r1.`comment` AS 'Comment' FROM (
SELECT r.itemID, r.rating_date_time, r.star, r.`comment`
FROM Rating r INNER JOIN Item i ON r.itemID = i.itemID
WHERE BINARY i.item_name='$ItemName') r1
INNER JOIN
(SELECT i1.itemID,
        CASE
                WHEN i1.cancelled_reason IS NULL THEN b1.username
                ELSE NULL
        END AS 'Winner'
FROM Item i1 INNER JOIN (
        SELECT b2.itemID, b2.username, b2.bid_amount FROM Bid b2 INNER JOIN (
                SELECT itemID, MAX(bid_amount) AS MaxBid FROM Bid GROUP BY
itemID
        ) AS MaxBids ON b2.itemID = MaxBids.itemID AND b2.bid_amount =
MaxBids.MaxBid
) b1 ON i1.itemID = b1.itemID
WHERE i1.auction_end_time<= NOW() AND b1.bid_amount >= i1.min_sale_price) r2
ON r1.itemID=r2.itemID
ORDER BY r1.rating_date_time DESC;
```

    - For each Rating from the sorted result:
        - Display the value of $Ratedby, $Date $Comment.
        - Display 5 Stars and color the number of stars equal to the value of $Star.
        - If the session variable '$UserType' value is 'AdminUser' then:
            - Display ***Delete*** link alongside the rating information.
    - Display **Close** button
- When Delete button is clicked against an item rating:
    - Run the **Delete Rating** subtask:
        - Delete the corresponding rating entry with $itemID from result

```
DELETE FROM Rating WHERE itemID= $itemID
```

    - Re-run the **View Item Ratings** task.
- When the **Close** button is clicked, go back to the **Item for Sale** form.

Edit Item Description

Abstract Code

- User clicked on the **Edit Description** link from the **Item for Sale** form.
- Display a popup window with:
    - *Add description* input field for adding new description ('$new_description') for the item.
    - **Save** and **Cancel** buttons.
- When **Save** button is clicked:

- Store the text from the *Add description* input field as $new_description of the Item that matches the session variable '$ItemID'.

```
UPDATE Item SET item_description = '$new_description'
WHERE itemID = '$ItemID'
```

- Go back to the **Item for Sale** form.

- When the **Cancel** button is clicked, go to the **Item for Sale** form.

Display Auction Results

Abstract Code

- Before trigger to generate auction result, proceed **Search Item** for any auction that does not have a declared winner but has already ended (based on the current time), then update those winners.
- User clicked on the ***View Auction Results*** button from the **Main Menu** form.
- Run the ***View Auction Results*** task: query for items whose auction has ended and display the results.

```
SELECT
        i.itemID AS `ID`,
         i.item_name AS `Item Name`,
        CASE WHEN i.cancelled_reason IS NULL THEN b.bid_amount ELSE NULL END AS `Sale Price`,
        CASE WHEN i.cancelled_reason IS NULL THEN b.username ELSE 'Cancelled' END AS `Winner`,
        i.auction_end_time AS `Auction Ended`
FROM Item i
LEFT JOIN (
        SELECT b1.itemID, b1.username, b1.bid_amount
        FROM Bid b1
        INNER JOIN (
                SELECT itemID, MAX(bid_amount) AS MaxBid
                FROM Bid
                GROUP BY itemID
        ) AS MaxBids ON b1.itemID = MaxBids.itemID AND b1.bid_amount = MaxBids.MaxBid
) b ON i.itemID = b.itemID
WHERE i.auction_end_time<= NOW() AND b.bid_amount >= i.min_sale_price OR b.bid_amount IS NULL
ORDER BY i.auction_end_time DESC;
```

- When any one of the item names is clicked:
  - Store the corresponding $itemID as session variable $ItemId
  - Store the $item_name as session variable '$ItemName'
  - Go to the **Display Item Results** form.
- When the ***Done*** button is clicked, go to the **Main Menu** form.

Display Item Results

Abstract Code

- User clicked on one of the ***Item Name*** link from the **Auction Results** form.

- Run the **View Item Results** task: query for details about the current item using the session variable $ItemId and display the results.

```
SELECT itemID, item_name, item_description, category_name, item_condition, returnable,
get_it_now_price, auction_end_time FROM Item WHERE itemID=$ItemID;
```

    - Enable link for proceeding with ***View Ratings***
- Run the **View Bid** task: query for latest four bids for the current item
    - Displayed with *BidAmount*, *TimeOfBid*, *Username*

```
(SELECT 'Cancelled' AS `Bid Amount`, auction_end_time AS `Time of Bid`,
'Administrator' AS `Username`  FROM Item WHERE itemID=$ItemID AND
cancelled_reason IS NOT NULL)
UNION ALL
(SELECT CAST(bid_amount AS CHAR) AS `Bid Amount`, time_of_bid AS `Time
of Bid`, username AS `Username`  FROM Bid WHERE itemID=$ItemID ORDER
BY bid_amount DESC LIMIT 4)
LIMIT 4;
```

- Wh\en the ***Cancel*** button is clicked, go to the **Auction Results** form.

Item Rating Submission

Abstract Code

- User clicked on the ***View Ratings*** link from the **Item Results** form.
- Run the **View Ratings** task to query all ratings related to current item:
    - Display the rating summary with $ItemID, $ItemName and Average Rating calculated using the below query:

```
SELECT
        ROUND(AVG(r.star), 1) AS `Average Rating`
FROM Rating r
INNER JOIN Item i ON r.itemID = i.itemID
WHERE BINARY i.item_name='$ItemName'
```

    - Display each rating comments and rated by user with below query:

```
SELECT r1.itemID, r2.Winner AS 'Rated By', r1. rating_date_time AS 'Date', r1.star
AS 'Star', r1.`comment` AS 'Comment' FROM (
SELECT r.itemID, r.rating_date_time, r.star, r.`comment`
FROM Rating r INNER JOIN Item i ON r.itemID = i.itemID
WHERE BINARY i.item_name='$ItemName') r1
INNER JOIN
(SELECT i1.itemID,
        CASE
                WHEN i1.cancelled_reason IS NULL THEN b1.username
                ELSE NULL
        END AS 'Winner'
FROM Item i1 INNER JOIN (
        SELECT b2.itemID, b2.username, b2.bid_amount FROM Bid b2 INNER JOIN (
                SELECT itemID, MAX(bid_amount) AS MaxBid FROM Bid GROUP BY
itemID
        ) AS MaxBids ON b2.itemID = MaxBids.itemID AND b2.bid_amount =
MaxBids.MaxBid
) b1 ON i1.itemID = b1.itemID
WHERE i1.auction_end_time<= NOW() AND b1.bid_amount >= i1.min_sale_price) r2
ON r1.itemID=r2.itemID
ORDER BY r1.rating_date_time DESC;
```

    - Display ***Close*** button.
- If the ratings result has a rating where ItemId value is same as the session variable '$ItemId':
    - Display ***Delete My Rating*** link against rating that was rated by the current user.
- Else there are no ratings added by the current user for the item won:
    - Run the **Add Item Ratings** task to add a new rating for the item won, display:
        - *My Rating* field with 5 empty Stars and an option to color any number of them ($star)
        - *Comments* input field for adding a new rating ($comment) for the item.
        - Display ***Close*** and ***Rate this Item*** buttons.
    - When ***Rate this Item*** button is clicked:
        - If the data validation is successful for *Comments* input field, then add a new entry in Rating with all details from input fields:

```
INSERT INTO Rating (itemID, rating_date_time, star, `comment`)
VALUES ($ItemID, NOW(), $star, '$comment');
```

        - Else *Comments* has invalid data, displaying **Rate Item** form with error message.
    - When ***Delete My Rating*** button is clicked:
        - Run the ***Delete Rating*** task to delete the entry from Rating.

```
DELETE FROM Rating WHERE itemID= $itemID
```

- When the ***Close*** button is clicked, go back to the **Item Results** form.

Display Category Information

Abstract Code

- Admin User clicked on the Category Reports button from the **Main Menu**.
- Run the Display Category Information task: Query the information from the Item and Category
    - Find the ***Items*** and the **Associated categories.**
    - Display Category Name as **Category**
    - Remove ***canceled auction items*** on the returned items and categories
    - Aggregate Get it Now Price to obtain **Min price, Max price, Avg Price**
        - If GetItNow price is not mentioned for any item, ignore that during aggregation.
    - Count the list of items and display as **Total Items**
    - Sort by Category Name in ascending order under **Category**
    - Show **Done** Button.
    - If the Admin user click on the **Done** button:
        - Return the Admin user to the **Main Menu**.

```
SELECT
   c.category_name AS `Category`,
   COUNT(i.itemID) AS `Total Items`,
   MIN(CASE WHEN i.get_it_now_price IS NOT NULL THEN i.get_it_now_price END) as `Min Price`,
   MAX(CASE WHEN i.get_it_now_price IS NOT NULL THEN i.get_it_now_price END) as `Max Price`,
   ROUND(AVG(CASE WHEN i.get_it_now_price IS NOT NULL THEN i.get_it_now_price END), 2) as
`Average Price`
FROM Category c LEFT JOIN Item i ON c.category_name = i.category_name
WHERE cancelled_username is null
GROUP BY c.category_name
ORDER BY c.category_name ASC;
```

Display User Information

Abstract Code

- Admin User clicked on the User Report button from the Main Menu
- Run the View User Reports task: Query the information from the User table, Items table, and the Rating table
    - Query
        - For all the users, count the listed Items and display as ***Listed.***
        - Total items sold by the user: For all the users, for each of the listed items with the winner in Auctions, count the items in Auction as ***Sold.***
        - Total items won by the user: For all the users listed as Winners in Auction, count the items, and display as ***Won.***
        - *Total number of items user rated: For all the users, count the items with ratings provided and display as* ***Rated.***
        - *Most Frequent Condition:*
            - *For all the users and their listed items, Identify the max of count of the Condition of items*
            - In the event of a tie, the lowest quality condition should be used.
            - If the user has no listings, should display as N/A
    - Show:
        - Buttons for ***Done.***

- If the Admin user click on **Done** button:
  - Return the Admin user to the **Main Menu**.

```
SELECT
        u.username AS `Username`,
        COUNT(DISTINCT i.itemID) AS `Listed`,
        COUNT(DISTINCT CASE WHEN i.auction_end_time <= NOW()
                AND (i.get_it_now_price IS NOT NULL AND b.bid_amount = i.get_it_now_price OR
b.bid_amount = max_bid.bid_amount)
                THEN i.itemID END) AS `Sold`,
        COUNT(DISTINCT CASE WHEN (i.get_it_now_price IS NOT NULL AND b.bid_amount =
i.get_it_now_price) OR b.bid_amount = max_bid.bid_amount THEN b.itemID END) AS `Won`,
        COUNT(DISTINCT r.itemID) AS `Rated`,
        COALESCE((
                SELECT i1.item_condition
                FROM Item i1
                WHERE i1.listeduser = u.username
                GROUP BY i1.item_condition
                ORDER BY COUNT(*) DESC,
                        FIELD(i1.item_condition, 'Poor', 'Fair', 'Good', 'Very Good', 'New') ASC
                LIMIT 1
        ), 'N/A') AS `Most Frequent Condition`

FROM User u

LEFT JOIN Item i ON u.username = i.listeduser
LEFT JOIN Bid b ON i.itemID = b.itemID
LEFT JOIN Rating r ON i.itemID = r.itemID
LEFT JOIN (
                SELECT itemID, MAX(bid_amount) AS bid_amount
                FROM Bid
                GROUP BY itemID
        ) AS max_bid ON i.itemID = max_bid.itemID

GROUP BY u.username;
```

Display Top Rated Items

Abstract Code

- Admin User clicked on the Top Rated Items hyperlink from the **Main Menu** form.
- Run the Display Top Rated Items task:
  - Query the information from the Item and Rating table.
    - Find the Items and the associated Ratings.
    - For each item in the result:
      - Display Item Name.
      - Find the average value of Star (rounded by one place) from the ratings for the current Item name and display as Average Rating.
      - Display count of ratings for the current Item name as Rating Count.
    - Sort the result by Average Rating in descending order.
    - List only top 10 Average Rating items.

```
SELECT
        i.item_name AS `Item Name`,
        ROUND(AVG(r.star), 1) AS `Average Rating`,
        COUNT(*) AS `Rating Count`
FROM Item i
LEFT JOIN Rating r ON i.itemID = r.itemID
WHERE r.star IS NOT NULL
GROUP BY item_name
ORDER BY 'Average Rating' DESC, item_name ASC
LIMIT 10;
```

- Show Done Button.
- If the Admin user click on Done button:
  - Return the Admin user to the Main Menu form.

Display Auction Statistics

Abstract Code

- Admin User clicked on the **Auction Statistics** hyperlink from the **Main Menu** form.
- Run the Display Auction Statistics task: Query the information from Rating
  - Query
    - **Auctions Active**: count of Actions where AuctionEnded is null in the Auctions

```
SELECT COUNT(auction_end_time) AS `Auctions Active`
FROM Item
WHERE auction_end_time > NOW() AND cancelled_username IS NULL ;
```

    - **Auctions Finished:** count of Actions where AuctionEnded is not null, and the Auction is not canceled.

```
SELECT COUNT(auction_end_time) AS `Auctions Finished`
FROM Item
WHERE auction_end_time <= NOW() AND cancelled_username IS NULL;
```

- **Auctions Won:** count of Actions where AuctionEnded is not null, and the Winner is not null

```
SELECT COUNT(DISTINCT i.itemID) as `Auctions Won`
FROM Item i
JOIN Bid b ON i.itemID = b.itemID
WHERE i.auction_end_time <= NOW() AND i.get_it_now_price <=
b.bid_amount AND i.cancelled_username IS NULL;
```

- **Auctions Canceled:** count of canceled Auctions

```
SELECT COUNT(itemID) as `Auctions Cancelled`
FROM Item
WHERE cancelled_username IS NOT NULL;
```

- **Items Rated*:*** count of items which are Rated

```
SELECT COUNT(DISTINCT itemID) AS `Items Rated` FROM Rating ;
```

- **Items not Rated*:*** count of items which are not rated

```
SELECT COUNT(DISTINCT itemID) as `Items Not Rated`
FROM Item
WHERE itemID NOT IN (SELECT DISTINCT itemID FROM Rating)
```

0r combine everything in ONE query

```
SELECT
   COUNT(CASE WHEN auction_end_time > NOW() AND cancelled_username IS NULL THEN 1 END) AS `Auctions Active`,
   (SELECT COUNT(auction_end_time) FROM Item i1 WHERE i1.auction_end_time <= NOW() AND i1.cancelled_username IS
NULL) AS `Auctions Finished`,
   (SELECT COUNT(DISTINCT i2.itemID) FROM Item i2 JOIN Bid b2 ON i2.itemID = b2.itemID WHERE i2.auction_end_time <=
NOW() AND i2.get_it_now_price <= b2.bid_amount AND i2.cancelled_username IS NULL) AS `Auctions Won`,
   COUNT(CASE WHEN cancelled_username IS NOT NULL THEN 1 END) AS `Auctions Cancelled`,
   COUNT(DISTINCT r.itemID) AS `Items Rated`,
   COUNT(DISTINCT CASE WHEN i.itemID NOT IN (SELECT DISTINCT itemID FROM Rating) THEN i.itemID END) AS `Items
Not Rated`
FROM Item i
LEFT JOIN Bid b ON i.itemID = b.itemID
LEFT JOIN Rating r ON i.itemID = r.itemID;
```

- Show:
  - Buttons for ***Done.***
  - If the Admin user click on Done button:
    - Return the Admin user to the main menu.

Display Canceled Auction Details

Abstract Code

- Admin User clicked on the **Canceled Auctions** hyperlink from the **Main Menu**.
- Run the View Auction Statistics task: Query the information from the Items table
  - Query: upon clicking:
    - ID: Display the ***Bid canceled*** items from the Auctions
    - Listed by: Display the user who listed the canceled auction items
    - Canceled Date: Display the auction canceled date
    - Reason : Display the reason for the auction cancellation
  - Show:
    - Buttons for ***Done.***
    - If the Admin user click on Done button:
      - Return the Admin user to the main menu.

```
SELECT
        COUNT(DISTINCT itemID) as `ID`,
        listeduser as `Listed by`,
        auction_end_time as `Cancelled Date` ,
        cancelled_reason as `Reason`
FROM Item
WHERE cancelled_username IS NOT NULL
GROUP BY itemID
ORDER BY itemID DESC;
```