Table of Contents

1 Abstract Code with SQL (PostgreSQL)

1.1 Log In

- User enters *username* ($username) and *password* ($password) input fields.
- If the user's record exists in the database, run a query to retrieve the user's $password.

```
SELECT password FROM "user" WHERE username= '$username';
```

- Confirm data validation. If user record exists and $password matches *password* input:
    - Data validation is successful and clicking the ***Login*** button presents the user with the **Main Menu** screen.
    - Else, the data validation is unsuccessful and an error message is displayed.
- Establish user role based on their user type and store as a session variable.
- Regardless of provided input, if the user clicks the Register button, the user is sent to the **Register** form.

1.2 Register

- User enters *First Name* ($first_name), *Last Name* ($last_name), *Username* ($username), *Password* ($password) and *Confirm Password* input fields.

```
INSERT INTO "user" (username, password, first_name, last_name, user_type)
SELECT '$username', '$password', '$first_name', '$last_name', ‘Regular’
WHERE NOT EXISTS (
   SELECT username FROM User WHERE username = '$username'
);
```

- User returns to the **Register** form for corrections and is presented with an informative error message, upon any of the following scenarios:
    - If the user does not fill all input fields.
    - *Username* input matches existing $username in database, and is therefore, not unique.
    - *Password* and *Confirm Password* inputs are not equal.
- Clicking the ***Cancel*** button returns the user to the **Login** form.

- If registration is successful (i.e, data validation is successful based on the above criteria and user does not encounter an error message), then:
    - Clicking the ***Register*** button sends the user to the **Main Menu** screen.
    - A new user record is created in the database. These users lack administrative privileges.

1.3 View Main Menu

- Run the **View Main Menu** task to query information about the user to determine whether or not they hold administrative privileges.

```
SELECT user_type FROM "user" WHERE username = $username;
```

- If they do not, show only ***Search for Items, List Item, View Auction Results*** tabs.

```
SELECT first_name, last_name FROM "user" WHERE username = $username;
```

- If they do, then in addition to the above, show their position ($position) and the following tabs: ***Category Report, User Report, Top Rated Items, Auction Statistics, Canceled Auction Details*** tabs.

```
SELECT first_name, last_name, position FROM "user" WHERE username = $username;
```

- User is able to select next action from the following choices:
    - Clicking the ***Search for Items*** button – jump to the **Search for Items** task.
    - Clicking the ***List Item*** button – jump to the **List Item** task.
    - Clicking the ***View Auction Results*** button – jump to the **View Auction Results** task.
    - Clicking the ***Category Report*** button – jump to the **View Category Report** task.
    - Clicking the ***User Report*** button – jump to the **View User Report** task.
    - Clicking the ***Top Rated Items*** button – jump to the **View Top Rated Items** task.
    - Clicking the ***Auction Statistics*** button – jump to the **View Auction Statistics** task.
    - Clicking the ***Canceled Auction Details*** button – jump to the **View Canceled Auction Details** task.

1.4 List Item

- User clicked on the ***List Item*** tab from the **Main Menu** screen.

- Run the **List Item** task. Query the $category_name, $condition and $auction_length information from the database to populate fixed drop down lists.

```
-- Retrieve item category
SELECT category_name AS item_category FROM category;

-- Retrieve item condition
SELECT enumlabel AS condition
FROM pg_enum
JOIN pg_type ON pg_enum.enumtypid = pg_type.oid
WHERE pg_type.typname = 'condition_enum'
ORDER BY pg_enum.enumsortorder;

-- Retrieve auction length
SELECT enumlabel AS auction_length
FROM pg_enum
JOIN pg_type ON pg_enum.enumtypid = pg_type.oid
WHERE pg_type.typname = 'auction_length_enum'
ORDER BY pg_enum.enumsortorder;
```

- Display the following text (with the exception of *Returns Accepted)* input fields and include their respective constraints:
    - *Item Name* ($name)
    - *Description* ($description)
    - *Starting Bid* ($starting_bid) – limited to two decimal places. Must be positive.
    - *Minimum Sale Price* ($min_sale_price)– limited to two decimal places. Must be positive.
    - *Get It Now Price* ($get_it_now_price)– limited to two decimal places. Must be positive. User may choose to leave this blank.
    - *Returns Accepted* ($is_returnable)– checkbox input field. User has the option to leave this unchecked.
- For successful verification, the form must follow the following constrictions:
    - All data fields except for *Get It Now Price* and *Returns Accepted* must be filled correctly. No error messages should occur.
    - *Get It Now Price* input must be higher than the *Starting Bid* input.
- Clicking the ***Cancel*** button returns the user to **Main Menu** form.
- If invalid data is entered, the item will not be listed. Instead, an error message will be displayed and the user will be returned to the **List Item** form to make corrections.

- Assuming successful data validation, clicking the ***List My Item*** button sends the user to the **View Item Listing** task.
    - Additionally, the item details will be added to the database along with a system generated unique $item_id.

```
INSERT INTO "item" (username, name, condition, category_name, description, starting_bid,
min_sale_price, get_it_now_price, end_time, is_returnable, cancellation_reason)
VALUES ($username, $name, $condition, $category_name, $description, $starting_bid,
$min_sale_price, $get_it_now_price, NOW() + INTERVAL '$auction_length DAY',
$is_returnable, NULL);
```

1.5 Search for Item

- User clicked on the ***Search for Item*** tab from the **Main Menu** screen.
- Run the **Search for Item** task. Query the $category_name and $condition information from the database to populate fixed drop down lists. These lists also contain blank rows in the case that the user does not want to include the search criteria.

```
-- item category
SELECT category_name AS item_category FROM category;

-- item condition
SELECT enumlabel AS item_condition
FROM pg_enum
JOIN pg_type ON pg_enum.enumtypid = pg_type.oid
WHERE pg_type.typname = 'condition_enum'
ORDER BY pg_enum.enumsortorder;
```

- Users have *Keyword, Minimum Price* and *Maximum Price* text input fields.
    - *Minimum Price* input must be lower than the *Maximum Price* input. If the user does not follow this restriction, display an error message once the user clicks the **Search** button.
- Clicking the **Cancel** button returns the user to the **Main Menu** screen.
- Clicking the **Search** button sends the user to the **Search Results** screen, only if:
    - User enters no criteria at all (i.e, does not provide any form input and does not choose any option from the drop down list(s)).

  - User enters multiple, valid search criteria. Only items that match all the criteria specified will be shown in the **Search Results** form.
- After the user inputs valid search criteria in the **Search for Item** form and clicks the ***Search*** button, run the **Search** task: query for information about the item.
  - Specifically, for each item, find the: $item_id, item $name, current $bid_amount, $username of the user with the highest bid, $get_it_now_price and the auction $end_time.
- Queried information is stored in a list and sorted by auction $end_time. Results are displayed with descending order of the earliest auction$end_time.
  - If there is no current $bid_amount, display dashes for Current Bid and Username fields instead.
  - If no $get_it_now_amount was specified, display dash instead.

```
SELECT
  i.item_id,
  i.name,
  COALESCE(MAX(b.bid_amount)::text, '--') AS current_bid,
  (
    SELECT b.username
    FROM "bid" b
    WHERE b.item_id = i.item_id
    ORDER BY b.bid_amount DESC
    LIMIT 1
  ) AS high_bidder,
  COALESCE(i.get_it_now_price::text, '--') AS get_it_now_price,
  i.end_time AS auction_ends
FROM
  "item" i
LEFT OUTER JOIN
  "bid" b ON i.item_id = b.item_id
GROUP BY
  i.item_id
HAVING
  (i.name LIKE '%' || $keyword || '%' OR $keyword IS NULL) OR
  (i.description LIKE '%' || $keyword || '%' OR $keyword IS NULL) AND
  (COALESCE(MAX(b.bid_amount), i.starting_bid) >= $min_price OR $min_price IS NULL)
AND
  (COALESCE(MAX(b.bid_amount), i.starting_bid) <= $max_price OR $max_price IS NULL)
AND
  (i.category_name = $category_name OR $category_name IS NULL) AND
  (i.condition <= $condition OR $condition IS NULL) AND
  i.end_time > NOW() AND
  i.cancellation_reason IS NULL
ORDER BY
  i.end_time ASC;
```

- Upon clicking an item name button, jump to that item's listing.
- Upon clicking the ***Back to Search*** button, return to the **Search for Item** form.

1.6 View Item

- User clicked on the ***Item Name*** tab from **Search Results** screen, or the ***List My Item*** button from the **List Item** form.
- Run the **View Item** task to query information about the item details, such as: its $item_id, $name, $description, $category_name, $condition, $is_returnable option, the $get_it_now_price, and the auction $end_time, as well as the bid history.
    - Display the above listed item details and populate the minimum bid amount below the *Your Bid* text input field.
        - If the $get_it_now_price was not provided by the user when listing the item, this field remains blank. Subsequently, the ***Get It Now*** button is not available for viewing.
        - If the $get_it_now_price was provided, run the **Bid/Buy Item** task.
            - Clicking the ***Get It Now*** button sets the current user as the auction winner. The auction end date is set to the date and time the ***Get It Now*** button was pressed. This information is updated and added to the database.

```
SELECT
  i.item_id AS "Item ID",
  i.name AS "Item Name",
  i.description AS "Description",
  i.category_name AS "Category",
  i.condition AS "Condition",
  i.is_returnable AS "Returns Accepted?",
  i.get_it_now_price AS "Get It Now Price",
  i.end_time AS "Auction Ends",
  COALESCE(MIN(b.bid_amount)::TEXT, '') AS "minimum bid"
FROM "item" i
LEFT JOIN "bid" b ON i.item_id = b.item_id
WHERE i.item_id = $item_id
GROUP BY i.item_id;
```

  - Display the bidding information if available, namely the: $bid_amount, $bid_time and the bidder's $username.
    - If no bids have been placed or item was canceled by the Admin user, the bid information remains unpopulated.
    - If bid history is available, store the above listed variables and sort by $bid_time. Display the top 4 bids in descending order of $bid_time, with the latest bid listed first.

```
SELECT
  bid_amount AS "Bid Amount",
  bid_time AS "Time of Bid",
  username AS "Username"
FROM bid
WHERE item_id = $item_id
AND bid_amount > 0
ORDER BY bid_amount DESC, bid_time DESC
LIMIT 4;
```

- Clicking the ***View Ratings*** button takes the user to the **<u>View Ratings</u>** screen and begins the **View Ratings** task.
- Users have a *Your Bid* text input field. If the current user's username matches the seller's username, then this input field will not be visible.
  - Input has to be at least $1 above the latest bid price. If the $get_it_now_price has been set, input must also fall below that price.
    - If the input isn't valid, display an error message.

  - If the user attempts to bid higher than the $get_it_now_price, an error message will be displayed that suggests the user clicks the ***Get It Now*** button instead of bidding.

```
-- check item pricing and bid info.
SELECT
   COALESCE(MAX(b.bid_amount), i.starting_bid) AS highest_bid,
   i.get_it_now_price
FROM "item" i
LEFT JOIN "bid" b ON i.item_id = b.item_id
WHERE i.item_id = '$item_id'
GROUP BY i.item_id, i.get_it_now_price;
```

- Run the **Edit Description** task; query the current user's information as well as the item's information. If the current user's $username and the seller's $username match, then this tab will be visible to them. If not, this tab will be invisible.
  - Clicking the ***Edit Description*** tab takes the user to the **Edit Description** window.
- Clicking the ***Close*** button returns the user to the **Search Results** screen.
- Administrative users are able to see the ***Cancel This Item*** button and run the **Cancel Auction** task. Clicking this button will provide a **Cancellation Reason** prompt.
- Assuming the *Your Bid* input is valid, clicking ***Bid On This Item*** button refreshes the current **Item Description** screen with the updated input. The user's bid is updated in the database and saved with the item details.

```
INSERT INTO Bid (username, item_id, bid_time, bid_amount)
VALUES ($username, $item_id, NOW(), $bid_amount);
```

1.7 Edit Description

- Current user's $username matched the seller's $username, which allowed this tab to be visible. If the $username of the two users had not matched, this tab would have remained invisible. User then clicked the ***Edit Description*** tab.

```
SELECT username
FROM "item"
WHERE item_id= $item_id;
```

- Run the **Edit Description** task. Query and display existing $description and provide a blank *New Description* input box.

```
SELECT description
FROM "item"
WHERE item_id= $item_id;
```

- Clicking the ***Save*** button overwrites the previous description, closes the window and returns the user to the **Item Listing** screen.

```
UPDATE "item"
SET description = $description
WHERE item_id= $item_id;
```

- Clicking the ***Cancel*** button returns the user to the **Item Listing** screen.

1.8 View Ratings

- User clicked on the ***View Ratings*** tab on the **Item Description** form.
- Run the **View Ratings** task; query information such as the $item_id, $name and its previous $number_of_stars.
    - Store the rating information (rater's $username, $number_of_stars given, $date_and_time and if applicable, $comment) in a list. Sort the list by rating $date_and_time. Display the ratings in descending order, with the most recent rating listed first.
    - If a comment was not provided, leave the comment area blank.
    - Calculate all previous $number_of_stars and display that as the average rating, rounded to one decimal place.
- Clicking the ***Close*** button returns the user to the **Item Description** form.

```
-- average rating
WITH ItemRatings AS (
  SELECT
    i.item_id,
    i.name,
      AVG(r.number_of_stars) OVER (PARTITION BY i.name) AS avg_rating
    FROM "item" i
    JOIN "rating" r ON i.item_id = r.item_id
    WHERE i.name = $name
)
SELECT
    item_id AS "Item ID",
    name AS "Item Name",
    ROUND(avg_rating, 1) AS "Average Rating"
FROM ItemRatings
WHERE item_id = $item_id
GROUP BY item_id, name, avg_rating;

-- user ratings
SELECT
    i.username AS "Rated by",
    r.number_of_stars,
    r.comments,
    r.date_and_time AS "Date"
FROM "rating" r
JOIN "item" i ON i.item_id = r.item_id
WHERE i.name = $name
ORDER BY r.date_and_time DESC;
```

1.9 Delete Rating

- User clicked on the **View Ratings** tab on the **<u>Item Description</u>** form.
- Run the **View Ratings** task; query information about the user ($username) and the item ($item_id).
    - If the user is an administrative user, they will also see a ***Delete*** tab on the top of each rating. By clicking, administrators remove the rating from the database and refresh the page.

```
-- check admin
SELECT usertype
```

```
FROM "user"
WHERE username = $username;

-- delete rating
DELETE FROM "rating"
WHERE item_id = $item_id;
```

- Clicking the ***Close*** button returns the user to the **Item Description** form.

1.10 View Auction Results

- User clicked on the ***View Auction Results*** button on the **Main Menu**.
- Run the **View Auction Results** task. Query item information regarding the $item_id, item $name, the final sale price, the $username of the person who won the bid for that item, and the auction $end_time.
    - Display $item_id and an ***Item Name*** tab to the item $name.
- For each item where the auction has ended, determine the sale price and winner.
    - If the item was purchased with the Get It Now price, set $min_sale_price to the $get_it_now_price.
    - If the highest bid at the end of the auction is ≥ $min_sale_price, set $min_sale_price to the highest bid amount.
    - If no bids were made, none met the $min_sale_price, or the item wasn't sold, then $min_sale_price will be left blank.
- Update the auction results depending on the scenarios above before displaying.
- Auction $end_time is used to display the results in descending order, with the most recently ended auction listed first.
- In the event that the auction was canceled by the administrator, the $min_sale_price will be updated to null, the auction $end_time will be updated to the time of cancellation.

```
SELECT
  i.item_id AS "ID",
  i.name AS "Item Name",
  CASE
    WHEN (mb.max_bid >= i.min_sale_price) AND i.cancellation_reason IS NULL
    THEN COALESCE(mb.max_bid::text, '--')
    WHEN (mb.max_bid < i.min_sale_price) AND i.cancellation_reason IS NULL THEN '--'
```

```
        WHEN mb.max_bid IS NULL AND i.cancellation_reason IS NULL THEN '--'
        WHEN i.cancellation_reason IS NOT NULL THEN '--' --auction got canceled
        ELSE '--'
    END AS "Sale Price",
    CASE
        WHEN mb.max_bid >= i.min_sale_price THEN (
            SELECT b.username
            FROM "bid" b
            WHERE b.item_id = i.item_id AND b.bid_amount = mb.max_bid
            ORDER BY b.bid_time DESC
            LIMIT 1
        )
        WHEN (mb.max_bid < i.min_sale_price) AND i.cancellation_reason IS NULL THEN '--'
        WHEN mb.max_bid IS NULL AND i.cancellation_reason IS NULL THEN '--'
        WHEN i.cancellation_reason IS NOT NULL THEN 'Canceled'
        ELSE '--'
    END AS "Winner",
    i.end_time AS "Auction Ended"
FROM "item" i
LEFT JOIN (
    SELECT
        item_id,
        MAX(bid_amount) AS max_bid
    FROM "bid"
    GROUP BY item_id
) AS mb ON i.item_id = mb.item_id
WHERE i.end_time <= NOW();
```

- Clicking on an ***Item Name***'s tab will take the user to that item's **Item Description** screen.
- Clicking the ***Done*** button returns the user to the **Main Menu** screen.

1.11 Rate Item

- User clicked on the ***Rate This Item*** button on the **Item Description** page, or the ***View Ratings*** tab on the **Item Results** page. The former is only possible if the user's username matches the username of the user who won the item and has not previously rated the item; the latter is available to all users.
- Run the **Rate Item** task: Query the $item_id, $name and calculate the average rating across $item_id, with their respective $number_of_stars, to view existing ratings.
- Run the **View Ratings** task: Query and display the $comments, rater's $username, rating's $date_and_time and $number_of_stars for all previous ratings made for the item.
- User may choose to add their own rating via the *Comments* text input box and by providing a *My Rating* star rating input.
    - The user's input for star rating and comment will be updated as their $number_of_stars and $comment, respectively.
    - Clicking the ***Rate This Item*** button will send the user to the **Item Rating** screen.

```
INSERT INTO "rating" (item_id, number_of_stars, comments, date_and_time)
VALUES ($item_id, $number_of_stars, $comments, NOW());
```

- If the user has previously rated, the *Comments* and the *My Rating* input field will be missing. Instead, run the **View Ratings** task.
    - If the user's $username matches the $username of a previous $comments, then the user will see a **Delete My Rating** tab as well.

```
SELECT i.username
FROM "item" i JOIN "rating" r ON i.item_id = r.item_id
WHERE r.item_id = $item_id;
```

- Clicking this tab deletes the rating from the database, refreshes the page and renders the user's $comments for that $item_id as blank. This also sends the user back to the **Rate Item** form.

```
DELETE FROM rating
WHERE item_id = $item_id;
```

- Upon clicking the ***Close*** button, return to the **Item Description** page.

1.12 Cancel Auction

- Admin selected the ***Cancel This Auction*** button on the **Item Description** page. Prior to clicking this button, the system confirmed that the user has an administrative position ($position).

```
SELECT user_type
FROM "user"
WHERE username = $username
```

- Run **Cancel Auction** task. Store the time when the administrative user clicked the button as auction's $end_time.
- User is prompted with a **Cancellation Reason Prompt** screen that contains a single *Cancellation Reason* text input field. If input is provided, store as $cancellation_reason. If no input is provided, a default value "Cancelled by Admin" will be used.
- A $bid_amount of $0 will be inputted when an auction is canceled.
- Regardless of whether or not the user filled in the input field, the user is able to click the ***Close*** button and is redirected back to the **Item Description** page.

```
INSERT INTO "bid" (username, item_ID, bid_time, bid_amount)
VALUES ($username, $item_id, NOW(), 0);


UPDATE "item"
SET
   end_time  = (SELECT MAX(bid_time) FROM "bid" WHERE item_id = $item_id),
   cancellation_reason = COALESCE($cancellation_reason, 'Cancelled by Admin')
WHERE item_id= $item_id;
```

1.13 Bid/Buy Item

- User clicked on the item tab from **Search Results.**
- Clicking the ***View Ratings*** tab takes the user to **Item Ratings.**
- Clicking the Edit ***Description*** button provides the user with a popup input field, provided the current user matches with the username of the listing user.

```
-- check current user and listing user
SELECT username FROM "item" WHERE item_id = $item_id;
```

  - User has the option to click the ***Close*** button to close the prompt, but can also provide input to replace the current description. If the user chooses to provide input, that input is stored as $description and is updated as that item's $description.

```
UPDATE "item"
SET description = $description
WHERE item_id= $item_id;
```

- Clicking the ***Get It Now*** button takes the user to the **Auction Result** page, as the user has immediately won the bid.
  - The user is now the winner of that $item_id.

```
-- Insert winning bid
INSERT INTO "bid" (username, item_id, bid_time, bid_amount)
VALUES ($username, $item_id, NOW(), (SELECT get_it_now_price FROM "item" WHERE
item_id = $item_id));

-- Update item end time
UPDATE "item"
SET end_time = (SELECT MAX(bid_time) FROM "bid" WHERE item_id = $item_id)
WHERE item_id = $item_id;
```

- Clicking the ***Close*** button returns the user to the **Search Results** form.
- Clicking the ***Bid On This Item*** button updates **Item for Sale** form with the most recent bid, provided the bid is valid.
  - System authenticates if data is valid if it is at least one dollar higher than the current highest bid and less than the $get_it_now_price.

```
-- current bid higher than max. bid and lower than get_it_now_price if exists
SELECT
   CASE
      WHEN $bid_amount >= COALESCE((SELECT MAX(bid_amount) FROM "bid"
WHERE item_id = $item_id), starting_bid - 1) + 1
      AND ($bid_amount < (SELECT COALESCE(get_it_now_price, $bid_amount + 1)
FROM "item" WHERE item_id = $item_id))
      THEN 'Valid'
      ELSE 'Invalid'
```

```
        END AS bid_status;
```

   - If the bid is not valid, the user is provided with an error message that suggests the user should click the ***Get It Now*** button instead of bidding again.

```
INSERT INTO "bid" (username, item_id, bid_time, bid_amount)
VALUES ($username, $item_id, NOW(), $bid_amount);
```

- The ***Cancel This Item*** button is available only to AdminUsers. When clicked, users are provided a **Cancellation Reason Prompt** form.

1.14 View Category Reports

- User clicked on the ***Category Report*** tab from the **Main Menu**. User is verified by the system to have an administrative position ($position).

```
SELECT user_type FROM "user" WHERE username = $username;
```

- Run the **View Category Report** task: query information about all items except for those who have $cancellation_reason. For each $category_name:
   - Count the number of unique $item_id. Display that number in the Total Items Row for that category.
   - Find the lowest $get_it_now_price across all the $item_id and display that number in the Min Price row.
   - Find the highest $get_it_now_price across all the $item_id and $category_name and display that number in the Min Price row.
   - Average the $get_it_now_price across all the $item_id excluding those that do not possess a $get_it_now_price.
   - Sort $category_name alphabetically
- Clicking the ***Done*** button returns the AdminUser to the **Main Menu.**

```
SELECT
  i.category_name AS "Category",
  COUNT(i.item_id) AS "Total Items",
  MIN(i.get_it_now_price) FILTER (WHERE i.get_it_now_price > 0) AS "Min Price",
  MAX(i.get_it_now_price) AS "Max Price",
  ROUND(AVG(i.get_it_now_price) FILTER (WHERE i.get_it_now_price > 0), 2) AS
"Average Price"
FROM "item" i
WHERE i.cancellation_reason IS NULL
GROUP BY i.category_name
ORDER BY i.category_name ASC;
```

1.15 View User Reports

- User clicked on the ***User Report*** tab from the **<u>Main Menu</u>**. User is verified by the system to have an administrative position ($position).

```
SELECT user_type FROM "user" WHERE username = $username;
```

- Run the **View User Report** task: query information about all users activities.
  - Aggregating the total number of items listed by each user, identifying unique $item_id and their $condition.
  - Summarizing the total number of auctions won.
  - Counting the total number of $comments each user has given and ensuring those entries are unique by focusing on the $date_and_time.
  - Determining the most frequent $condition of items listed by each user by selecting the lowest-quality condition in the event of a tie.
  - If the user has no listing, display "N/A" for the most frequent condition.
- Format results to display each user's username along with the aggregated statistics: the total number of items listed, items sold, auctions won, ratings given, and the predominant condition of their listings.
- Sort results by the total number of listings per user in descending order.
- Clicking the ***Done*** button returns the AdminUser to the **<u>Main Menu</u>**.

```
WITH UserListings AS (
  SELECT
    username,
    COUNT(*) AS total_listings,
    COUNT(DISTINCT item_id) AS unique_listings
  FROM "item" i
  GROUP BY username
), UserSold AS (
  SELECT
    i.username,
    COUNT(*) AS total_sold
  FROM "item" i
  JOIN "bid" b ON b.item_id = i.item_id
  WHERE b.bid_amount >= i.min_sale_price AND
  i.end_time <= NOW() AND
  i.cancellation_reason IS NULL
  GROUP BY i.username
), UserWins AS (
  SELECT
    b.username,
    COUNT(*) AS wins
  FROM "bid" b
  JOIN "item" i ON b.item_id = i.item_id
  WHERE
  b.bid_amount >= i.min_sale_price AND
  b.bid_time = (
   SELECT MAX(bid_time)
   FROM "bid"
   WHERE item_id = b.item_id
  ) AND
  i.cancellation_reason IS NULL
  GROUP BY b.username
), UserComments AS (
  SELECT
    username,
    COUNT(DISTINCT date_and_time) AS total_comments
  FROM "rating"
  JOIN "item" ON item.item_id = rating.item_id
  GROUP BY username
), MostFrequentCondition AS (
  SELECT
    cc.username,
    COALESCE(cc.condition::text, 'N/A') AS most_frequent_condition
    FROM (
        SELECT DISTINCT ON (username) username, condition
        FROM (
            SELECT
                username,
                condition,
                COUNT(*) AS condition_count
            FROM "item"
            GROUP BY username, condition
            ORDER BY username, condition_count DESC
        ) AS condition_counts
    ) AS cc
)
SELECT
    u.username AS "Username",
    COALESCE(ul.total_listings, 0) AS "Listed",
    COALESCE(us.total_sold, 0) AS "Sold",
    COALESCE(uw.wins, 0) AS "Won",
    COALESCE(uc.total_comments, 0) AS "Rated",
    COALESCE(mfc.most_frequent_condition::text, 'N/A') AS "Most Frequent Condition"
FROM "user" u
LEFT JOIN UserListings ul ON u.username = ul.username
LEFT JOIN UserSold us on u.username = us.username
LEFT JOIN UserWins uw ON u.username = uw.username
LEFT JOIN UserComments uc ON u.username = uc.username
LEFT JOIN MostFrequentCondition mfc ON u.username = mfc.username
ORDER BY total_listings DESC;
```

1.16 View Top Rated Items

- User clicked on the ***Top Rated Items Report*** tab from the **Main Menu**. User is verified by the system to have an administrative position ($position).

```
SELECT user_type FROM "user" WHERE username = $username;
```

- Run the **View Top Rated Items Report** task: query information about top 10 rated items within the database. Specifically,
    - Iterate through the $item_id and their $number_of_stars. Skip items that do not have $number_of_stars. Use each item's $number_of_stars to determine an average rating and store that in a list.
    - For each item in that list, count the total number of $number_of_stars.
    - Display those items by descending $number_of_stars ranking.
        - In cases where items have identical average ratings, they are then ordered by item $name in ascending order.
- Clicking the ***Done*** button returns the AdminUser to the **Main Menu**.

```
SELECT
  i.name AS "Item Name",
  ROUND(AVG(r.number_of_stars)::NUMERIC, 1) AS "Average Rating",
  COUNT(r.number_of_stars) AS "Rating Count"
FROM "rating" r
JOIN "item" i ON r.item_id = i.item_id
GROUP BY i.name
HAVING COUNT(r.number_of_stars) > 0
ORDER BY "Average Rating" DESC, "Item Name"
LIMIT 10;
```

1.17 View Auction Statistics

- User clicked on the Auction ***Statistics Report*** tab from the **Main Menu**. User is verified by the system to have an administrative position ($position).

```
SELECT user_type FROM "user" WHERE username = $username;
```

- Run the **View Auction Statistics** task: query information about all auctions and items within the database.
    - Retrieve the total count of active auctions and $end_time is greater than the current time.
    - Summarize completed auctions where $end_time has passed.
    - Count the number of won auctions by checking for entries that have not been canceled by the administrator.

  - Gather the number of canceled auctions that have not been canceled.
  - Aggregate the total number of rated items by counting $comments associated with that $username. Do not include those that do not possess $number_of_stars.
- Clicking the ***Done*** button returns the AdminUser to the **<u>Main Menu</u>**.

```
-- Active Auctions
SELECT COUNT(*) AS "Auctions Active"
FROM "item"
WHERE cancellation_reason IS NULL AND end_time > NOW();


-- Completed Auctions
SELECT COUNT(*) AS "Auctions Finished"
FROM "item"
WHERE cancellation_reason IS NULL AND end_time <= NOW();

-- Won Auctions
SELECT COUNT(DISTINCT i.item_id) AS "Auctions Won"
FROM "bid" b
JOIN "item" i ON b.item_id = i.item_id
WHERE i.cancellation_reason IS NULL AND i.end_time <= NOW()
AND (SELECT MAX(bid_amount) FROM "bid" WHERE $item_id = item_id) >=
i.min_sale_price;

-- Canceled Auctions
SELECT COUNT(*) AS "Auctions Cancelled"
FROM "item"
WHERE cancellation_reason IS NOT NULL;

-- Rated Items
SELECT COUNT(*) AS "Items Rated"
FROM "rating";

-- Unrated Items
SELECT COUNT(*) AS "Items Not Rated"
FROM "item" i
LEFT JOIN "rating" r ON i.item_id = r.item_id
```

```
WHERE r.item_id IS NULL;
```

1.18 View Cancelled Auction Details

- User clicked on the ***Cancelled Auction Details Report*** tab from the **Main Menu**. User is verified by the system to have $position.

```
SELECT user_type FROM "user" WHERE username = $username;
```

- Run the **View Cancelled Auction Details** task: query database for all auctions canceled by AdminUsers.
    - For each auction that has been canceled, display the $item_id, the $username of the associated user, the $cancellation_reason, and the auction's $end_time .
    - Displayed items are sorted by descending $item_id.
- Clicking the ***Done*** button returns the AdminUser to the **Main Menu**.

```
SELECT
  i.item_id AS "ID",
  i.username AS "Listed By",
  i.end_time AS "Cancelled Date",
  i.cancellation_reason AS "Reason"
FROM "item" i
WHERE i.cancellation_reason IS NOT NULL
ORDER BY i.item_id DESC;
```

1.19 View Item Listing/Results

- User clicked on the ***Item Name*** tab from the **Auction Results** page. User is verified by the system to have an administrative position ($position).

```
SELECT user_type FROM "user" WHERE username = $username;
```

- Run the **View Item Listing/Results** task: query information about the item that the user clicked.
    - For the item, display the $item_id, item $name, $description, $category_name, $condition, $is_returnable, $get_it_now_price, and query the auction's $end_time.

```
SELECT
        item_id AS "Item ID",
        name AS "Item Name",
        description AS "Description",
        category_name AS "Category",
        condition AS "Condition",
        is_returnable AS "Returns Accepted",
        get_it_now_price AS "Get It Now Price",
        end_time AS "Auction Ended"
FROM "item"
WHERE item_id = $item_id;
```

- Query bid history, display $bid_amount, $bid_time, and the bidder's $username for all bids on the item, sorted by $bid_time (most recent to oldest).

```
SELECT
         CASE
               WHEN bid_amount = 0 THEN 'Cancelled'
               ELSE bid_amount::text
         END AS "Bid Amount",
         bid_time AS "Time of Bid",
         username AS "Username"
FROM bid
WHERE item_id = $item_id
ORDER BY bid_time DESC
LIMIT 4;
```

- For auction winners, provide functionality to rate the item and view previous ratings.
    - Clicking the ***View Rating*** button sends the user to the **Item Rating Form**.
- Clicking the ***Close*** button returns the user to the **Search Result** screen.