Table of Contents

Comment

(1) We tested SQL code with MYSQL.
(2) We write the SQL code in an one-stop manner: request relevant information, calculate intermediate variables, organize and rename columns in a single (and in many cases long) SQL query for each subtask

General Rules

We follow the rules in the provided example

- Form: **Form**
- Button: ***Button***
- Task: **Task**
- Input Fields: *Input* (and associated input content @Input)
- Database.Table.Attribute: Database.Table.Attribute[1][2]
- Input content typed in the input field, or saved session variable (such as current user): @Input, @UserID
- Strings: 'String'
- Tab: **"Tab"**
- MYSQL functon or user defined function: FUNCTION()
- MYSQL keyword: SELECT
- MYSQL generated input filled (require user input): "$Input_Field"
- MYSQL comment: -- comment

[1] Note: only do this in main text but not MYSQL code as otherwise almost everything will be yellow (everything except function, keyword, local variables and delimiters in code is either database name or table name or column name)
[2] The name of the database is optional and our database is named as "gameswapDB"

Since we need to calculate distance between two postal codes multiple times and each time it involves exactly the same large body of operations, we define a function `cal_dist(postal_code1, postal_code2)' separately and call this function whenever appropriate.

```
DROP FUNCTION IF EXISTS cal_dist;
DELIMITER $$
CREATE FUNCTION cal_dist(
   zip1 CHAR(5),
   zip2 CHAR(5)
) RETURNS float
READS SQL DATA
BEGIN
   DECLARE lat1,lon1,lat2,lon2 FLOAT DEFAULT 0;
   DECLARE delta_lat FLOAT DEFAULT 0;
   DECLARE delta_lon FLOAT DEFAULT 0;
   DECLARE a FLOAT DEFAULT 0;
   DECLARE c FLOAT DEFAULT 0;

   SELECT latitude,longitude
   INTO lat1,lon1
   FROM gameswapDB.Location
   WHERE postalCode=zip1;

   SELECT latitude,longitude
   INTO lat2,lon2
   FROM gameswapDB.Location
   WHERE postalCode=zip2;


   SELECT pi()*lat1/180-pi()*lat2/180 INTO delta_lat;
   SELECT pi()*lon1/180-pi()*lon2/180 INTO delta_lon;
   SELECT power(sin(delta_lat/2),2)+cos(pi()*lat1/180)*cos(pi()*lat2/180)*
          power(sin(delta_lon/2),2) INTO a;
   SELECT 2*atan2(sqrt(a),sqrt(1-a)) INTO c;
   RETURN c*6371;
END$$
DELIMITER ;
```

Login

Abstract Code

- If ***Register*** button is clicked:
  - Go to **Registration** form
- User enters *email/phone* @Email_or_Phone and *password* @Password into input fields
- If data validation is successful for both *email/phone* and *password* input field, then
  - When ***Login*** button is clicked:
    - If "@" in @Email_or_Phone:
      - If ("$Email_or_Phone" not found in User.email) or (password incorrect):

```
SELECT password FROM gameswapDB.User WHERE User.email=@Email_or_Phone;
```

        - Go back to **Login** form, with error message
      - Else
        - Store login information as session variable @UserID
        - Go to **Main Menu**
    - Else:
      - If (@Email_or_Phone not found in Phone.phoneNumber) or (password incorrect):

```
SELECT password FROM gameswapDB.User WHERE User.email=(
  SELECT ownerEmail FROM gameswapDB.Phone WHERE Phone.phoneNumber=
@Email_or_Phone);
```

        - Go back to **Login** form, with error message
      - Else
        - Store login information as session variable @UserID
        - Go to **Main Menu**

Registration

Abstract Code

- 1 If data validation is successful for all fields (*Email, Nick_Name, Password, City, First_Name, State, Last_Name, Postal_Code, Phone_Number, Phone_Type, Share_Phone*), then
  - When ***Register*** button is clicked:
    - If @Email is found in User.email:

```
SELECT COUNT(*) FROM gameswapDB.User WHERE User.email=@Email;
```

      - Go back to **Register** form, with error message “Email already used”
    - Else if @Phone_Number is found in Phone.phoneNumber:

```
SELECT COUNT(*) FROM gameswapDB.Phone WHERE  hone.phoneNumber=@Phone_Number;
```

      - Go back to **Register** form, with error message “Phone already used”
    - Else if @Postal_Code is not found in Location.postalCode:

```
SELECT COUNT(*) FROM gameswapDB.Location WHERE  Location.postalCode=@Postal_Code;
```

      - Go back to **Register** form, with error message “Please enter valid Postal Code”
    - Else if @Postal_Code is found in Location.postalCode but city, state info are wrong

```
SELECT city, state FROM gameswapDB.Location WHERE Location.postalCode=@Postal_Code;
```

      - Go back to **Register** form, with error message and suggestion for City and State info
    - Else:
      - Store user information in User, Phone

```
INSERT INTO User(email,postalCode,password,first_name,last_name,nick_name)
VALUES (@Email,@Postal_Code,@Password,@First_Name,@Last_Name,@Nick_Name);
INSERT INTO gameswapDB.Phone(phoneNumber,ownerEmail,phone_type,share_phone)
VALUES(@Phone_Number,@Email,@Phone_Type,@Share_Phone);
```

      - Go to **Login** form

Main/Navigation Menu

Abstract Code

- Show **"My Rating", "Unaccepted swaps", "Unrated Swaps"** tab and ***Log out, List Item, My Items, Search items, Swap history, Update my info*** button
- Query and show first_name, last_name in User where User.email=="$UserID", show welcome information

```
SELECT first_name,last_name FROM gameswapDB.User WHERE User.email=@UserID;
```

- Join SwapRecord and Item table and calculate the number of unaccepted swaps[3], show in **"Unaccepted swaps"**, show color

```
SELECT COUNT(*)
FROM gameswapDB.SwapRecord NATURAL JOIN (
   SELECT ownerEmail AS counterpartyEmail, ItemID AS desiredItemID
   FROM gameswapDB.Item
) AS M
WHERE M.counterpartyEmail=@UserID AND SwapRecord.status IS NULL;
```

- Join SwapRecord with Item Table (join twice for desiredItem and proposedItem separately) and calculate the number of unrated swaps, show in **"unrated swaps"**, show color

```
SELECT COUNT(*)
FROM gameswapDB.SwapRecord NATURAL JOIN (
        SELECT ownerEmail AS proposerEmail, ItemID AS proposedItemID
        FROM gameswapDB.Item
)AS M1 NATURAL JOIN (
        SELECT ownerEmail AS counterpartyEmail, ItemID AS desiredItemID
        FROM gameswapDB.Item
)AS M2
WHERE ((proposerEmail=@UserID AND proposer_rate IS NULL) OR
(counterpartyEmail=@UserID AND counterparty_rate IS NULL)) AND (status=1);
```

- (Join SwapRecord and Item Table (as proposedItem), select proposer_rate where proposerEmail=@UserID), Union with (Join SwapRecord and Item Table (as desiredItem), select counterparty_rate where counterpartyEmail=@UserID) calculate the average of the result column and shown in **"my rating"**[4],

[3] Here For unaccepted swaps, we only focus on those swaps that the current user appear as a counterparty

[4] Here we assume the rating of a user is the average rating given by his/her counterparty in a swap (i.e. how other people rate him/her), instead of how he/she rate other people

```
SELECT AVG(rate)
FROM(
   -- when I am the proposer of the swap, select the counterparty_rate
   SELECT counterparty_rate AS rate
   FROM gameswapDB.SwapRecord
      NATURAL JOIN(
         SELECT ownerEmail AS proposerEmail, ItemID AS proposedItemID
               FROM gameswapDB.Item
      ) AS M1
   WHERE M1.proposerEmail=@UserID
   UNION
   -- when I am the counterparty of the swap, select the proposer_rate
        SELECT proposer_rate AS rate
   FROM gameswapDB.SwapRecord
      NATURAL JOIN(
         SELECT ownerEmail AS counterpartyEmail, ItemID AS desiredItemID
               FROM gameswapDB.Item
      ) AS M2
   WHERE M2.counterpartyEmail=@UserID
) AS M;
```

- Upon
  - Click ***Logout*** button --- Jump to the **Login** task
  - Click ***List Item*** button --- Jump to the **List Item** task
  - Click ***My items*** button --- Jump to the **View My items** task
  - Click ***Search items*** button --- Jump to the **Search items** task
  - Click ***Swap history*** button --- Jump to the **Swap history** task
  - Click ***Update my info*** button --- Jump to the **Update User info** task
  - Click Unaccepted swaps link
    - If the number in **"Unaccepted swaps"**>0
      - Jump to the **Accept/Reject Swap** task
    - Else
      - Stay in the **Main** menu
  - Click "Unrated swaps" button
    - If the number in **"Unrated swaps">**0
      - Jump to the **Rate Swaps** task
    - Else
      - Stay in the **Main** menu

List Item

Abstract Code

- User clicked on ***List item*** button from **Main** Menu
- If the number in **"Unaccepted swaps"**>5 or the number in **"Unrated swaps"**>2:
  - Show error message
- Else
  - Run the **List item** task, show *GameType, Title, Condition, Description* fields
  - If @Game_Type == "Jigsaw puzzle"
    - Show *PieceCount* field
  - Else if @Game_Type == "Video game"
    - Show *Platform, Media* field
  - Else if @Game_Type == "Computer game"
    - Show *Platform* field
  - If data validation is successful for all fields, then
    - When ***List Item*** button is clicked
      - Store item information in Item

```
INSERT INTO Item(ownerEmail, title, description, game_condition)
VALUES (@Email,@Title,@Description,@Game_Condition);
```

      - If @Game_Type == "Board Game" : Insert into BoardGame

```
INSERT INTO Boardgame(itemID) VALUES (LAST_INSERT_ID());
```

      - Else if @Game_Type == "Card Game" : Insert into CardGame

```
INSERT INTO Cardgame(itemID) VALUES (LAST_INSERT_ID());
```

      - Else if @Game_Type == "Jigsaw Puzzle" : Insert into JigsawPuzzle

```
INSERT INTO Jigsawpuzzle(itemID, piece  count) VALUES (LAST  INSERT  ID(),@Piece  Count);
```

      - Else if @Game_Type== "Video Game" : Insert into VideoGame

```
INSERT INTO Videogame(itemID, platformID, media)
VALUES (LAST_INSERT_ID(),
    (SELECT PlatformID FROM VideoPlatform WHERE platform_name=@Video_Platform),
@Media);
```

      - Else if @Game_Type == "Computer Game" : Insert into ComputerGame

```
INSERT INTO Computergame(itemID, computer_platform) VALUES
LAST_INSERT_ID(),@Computer_Platform);
```

      - Show item insertion succeeds information
      - If ***OK*** button is clicked
      - Go to **Main** Menu

View My Items

Abstract Code

- User clicked on ***My Items*** button from **Main** Menu:
- Run the **View My Items** task: query Item Table (and subcategory tables) where (item.ownerEmail==@UserID)

```
SELECT itemID, CASE
WHEN Item.itemID IN (SELECT itemID FROM BoardGame) THEN 'Board Game'
WHEN Item.itemID IN (SELECT itemID FROM CardGame) THEN 'Card Game'
WHEN Item.itemID IN (SELECT itemID FROM JigsawPuzzle) THEN 'Jigsaw Puzzle'
WHEN Item.itemID IN (SELECT itemID FROM ComputerGame) THEN 'Computer Game'
WHEN Item.itemID IN (SELECT itemID FROM VideoGame) THEN 'Video Game'
ELSE 'Error Type'
END AS Game_Type, title, game_condition, description
FROM Item WHERE Item.ownerEmail=@UserID;
```

- Count and display items owned by current user by game type

```
SELECT
(SELECT COUNT(*) FROM Item WHERE Item.ownerEmail=@UserID AND Item.itemID IN
(SELECT itemID FROM BoardGame)) AS BoardGame,
(SELECT COUNT(*) FROM Item WHERE Item.ownerEmail=@UserID AND Item.itemID IN
(SELECT itemID FROM CardGame)) AS CardGame,
(SELECT COUNT(*) FROM Item WHERE Item.ownerEmail=@UserID AND Item.itemID IN
(SELECT itemID FROM ComputerGame)) AS ComputerGame,
(SELECT COUNT(*) FROM Item WHERE Item.ownerEmail=@UserID AND Item.itemID IN
(SELECT itemID FROM JigsawPuzzle)) AS JigsawPuzzle,
(SELECT COUNT(*) FROM Item WHERE Item.ownerEmail=@UserID AND Item.itemID IN
(SELECT itemID FROM VideoGame)) AS VideoGame,
(SELECT COUNT(*) FROM Item WHERE Item.ownerEmail=@UserID) AS Total;
```

Search Items

Abstract Code

- A ratio group (i.e. checkbox that can only select a single choice) (with four options: *By keyword, In my postal code, Within miles, In postal code*) for @SearchType with parameter *@SearchParam* (call it @Distance, @PostalCode, @StringSearch depending on corresponding @SearchType) associates with all four options except *In my postal code* shows up when user clicked on ***My Item*** button from **Main** Menu
- When ***Search*** button if clicked
  - If data validation is successful (@SearchType is available and @SearchParam is available when necessary)
    - If @SearchType="By keyword":
      - Query Item Table (Title, game_condition, description columns) and select item with @StringSearch[5], ownerEmail!=@UserID, ItemID not appear in (SwapRecord.proposedItemID, SwapRecord.desiredItemID where SwapRecord.status==1 or IS NULL), join with User Table and calculate distance, order by distance

```
SELECT Item.itemID AS 'Item', CASE
WHEN Item.itemID IN (SELECT itemID FROM BoardGame) THEN 'Board Game'
WHEN Item.itemID IN (SELECT itemID FROM CardGame) THEN 'Card Game'
WHEN Item.itemID IN (SELECT itemID FROM JigsawPuzzle) THEN 'Jigsaw Puzzle'
WHEN Item.itemID IN (SELECT itemID FROM ComputerGame) THEN 'Computer Game'
WHEN Item.itemID IN (SELECT itemID FROM VideoGame) THEN 'Video Game'
ELSE 'Error Type'
END AS  'Game type',
    Item.Title AS 'Title',
    Item.game_condition AS 'Condition',
    Item.Description AS 'Description',
    cal_dist((SELECT User.postalCode FROM User WHERE User.email =@UserID),
User.postalCode) AS 'Distance'
FROM Item INNER JOIN User ON User.Email = Item.ownerEmail
WHERE User.Email != @UserID
 AND (Item.Description LIKE @StringSearch OR Item.Title LIKE @StringSearch)
 AND (Item.itemID NOT IN (
  SELECT proposedItemID FROM SwapRecord WHERE status = 1 OR status IS NULL))
 AND (Item.itemID NOT IN (
  SELECT desiredItemID FROM SwapRecord WHERE status = 1 OR status IS NULL))
ORDER BY Distance ASC, Item.itemID ASC;
```

      - Else if @SearchType="In my postal code":

[5] To use LIKE function in MYSQL, we need to add % to the beginning and end of the input string

- Join Item and User Table and select item with users in the same postalCode as the postalCode of @UserID and ownerEmail!=@UserID, ItemID not appear in (SwapRecord.proposedItemID, SwapRecord.desiredItemID where SwapRecord.status==1 or IS NULL), calculate and order by distance

```
SELECT Item.itemID AS 'Item', CASE
   WHEN Item.itemID IN (SELECT itemID FROM BoardGame) THEN 'Board Game'
   WHEN Item.itemID IN (SELECT itemID FROM CardGame) THEN 'Card Game'
   WHEN Item.itemID IN (SELECT itemID FROM JigsawPuzzle) THEN 'Jigsaw Puzzle'
   WHEN Item.itemID IN (SELECT itemID FROM ComputerGame) THEN 'Computer Game'
   WHEN Item.itemID IN (SELECT itemID FROM VideoGame) THEN 'Video Game'
   ELSE 'Error Type'
   END AS  'Game type',
     Item.Title AS 'Title',
     Item.game_condition AS 'Condition',
     Item.Description AS 'Description',
     cal_dist((SELECT User.postalCode FROM User WHERE User.email = @UserID),
User.postalCode) AS 'Distance'
FROM Item INNER JOIN User ON User.Email = Item.ownerEmail
WHERE User.postalCode = (SELECT User.postalCode FROM User WHERE User.email =
@UserID)
  AND User.Email != @UserID
  AND (Item.itemID NOT IN (
    SELECT proposedItemID FROM SwapRecord WHERE status = 1 OR status IS NULL))
  AND (Item.itemID NOT IN (
    SELECT desiredItemID FROM SwapRecord WHERE status = 1 OR status IS NULL))
ORDER BY Distance ASC, Item.itemID ASC;
```

- Else if @Search Type==”In postal code”
  - Query User Table and obtain users in @PostalCode, join with Item Table, select items with ownerEmail!=@UserID, ItemID not appear in (SwapRecord.proposedItemID, SwapRecord.desiredItemID where SwapRecord.status==1 or IS NULL), calculate and order by distance

```
SELECT Item.itemID AS 'Item', CASE
   WHEN Item.itemID IN (SELECT itemID FROM BoardGame) THEN 'Board Game'
   WHEN Item.itemID IN (SELECT itemID FROM CardGame) THEN 'Card Game'
   WHEN Item.itemID IN (SELECT itemID FROM JigsawPuzzle) THEN 'Jigsaw Puzzle'
   WHEN Item.itemID IN (SELECT itemID FROM ComputerGame) THEN 'Computer Game'
   WHEN Item.itemID IN (SELECT itemID FROM VideoGame) THEN 'Video Game'
   ELSE 'Error Type'
   END AS  'Game type',
     Item.Title AS 'Title',
     Item.game_condition AS 'Condition',
     Item.Description AS 'Description',
     cal_dist((SELECT User.postalCode FROM User WHERE User.email = @UserID),
User.postalCode) AS 'Distance'
FROM Item INNER JOIN User ON User.Email = Item.ownerEmail
WHERE User.postalCode =@PostalCode AND User.Email != @UserID
  AND (Item.itemID NOT IN (
    SELECT proposedItemID FROM SwapRecord WHERE status = 1 OR status IS NULL))
  AND (Item.itemID NOT IN (
    SELECT desiredItemID FROM SwapRecord WHERE status = 1 OR status IS NULL))
ORDER BY Distance ASC, Item.itemID ASC;
```

- Else if @SearchType==”Within miles”:
  - Query Location Table and obtain a list of postalCode whose distance to the postal code of @UserID is less than @Distance, Query User Table where the postal code of users are in the proceeding list. Join with Item table and select items where ownerEmail!=@UserID, ItemID not appear in (SwapRecord.proposedItemID, SwapRecord.desiredItemID where SwapRecord.status==1 or IS NULL). Calculate and order by distance

```
SELECT Item.itemID AS 'Item',CASE
   WHEN Item.itemID IN (SELECT itemID FROM BoardGame) THEN 'Board Game'
   WHEN Item.itemID IN (SELECT itemID FROM CardGame) THEN 'Card Game'
   WHEN Item.itemID IN (SELECT itemID FROM JigsawPuzzle) THEN 'Jigsaw Puzzle'
   WHEN Item.itemID IN (SELECT itemID FROM ComputerGame) THEN 'Computer Game'
   WHEN Item.itemID IN (SELECT itemID FROM VideoGame) THEN 'Video Game'
   ELSE 'Error Type'
   END AS  'Game type',
     Item.Title AS 'Title',
     Item.game_condition AS 'Condition',
     Item.Description AS 'Description',
     cal_dist((SELECT User.postalCode FROM User WHERE User.email = @UserID),
User.postalCode) AS 'Distance'
FROM Item INNER JOIN User ON User.Email = Item.ownerEmail
WHERE User.postalCode IN (
   SELECT User.postalCode
   FROM User INNER JOIN Item ON User.Email = Item.ownerEmail
   WHERE cal_dist((SELECT User.postalCode FROM User WHERE User.email = @UserID),
User.postalCode) <= @Distance
)
AND User.Email != @UserID
AND (Item.itemID NOT IN (
  SELECT proposedItemID FROM SwapRecord WHERE status = 1 OR status IS NULL))
AND (Item.itemID NOT IN (
  SELECT desiredItemID FROM SwapRecord WHERE status = 1 OR status IS NULL))
ORDER BY Distance ASC, Item.itemID ASC;
```

- If length of items>0:
  - Display query results
- Else:
  - Return to the search form with error message “Sorry, no results found!”

View Item

Abstract Code

- Upon user clicked detail link from item lists (save @ItemID from the click operation)
- Query Item Table (and subcategory tables), join with User Table, calculate the rating of the owner of the item (almost the same as my rating subtask in main menu task)

```
SELECT itemID, CASE
WHEN Item.itemID IN (SELECT itemID FROM BoardGame) THEN 'Board Game'
WHEN Item.itemID IN (SELECT itemID FROM CardGame) THEN 'Card Game'
WHEN Item.itemID IN (SELECT itemID FROM JigsawPuzzle) THEN 'Jigsaw Puzzle'
WHEN Item.itemID IN (SELECT itemID FROM ComputerGame) THEN 'Computer Game'
WHEN Item.itemID IN (SELECT itemID FROM VideoGame) THEN 'Video Game'
ELSE 'Error Type'
END AS Game_Type, title, game_condition, description, first_name, last_name,
cal_dist(postalCode, (SELECT postalCode FROM User WHERE email=@UserID)) AS
Distance,
(SELECT AVG(rate)
FROM(
   SELECT proposerEmail as Email, counterparty_rate AS rate
   FROM gameswapDB.SwapRecord
      NATURAL JOIN(
         SELECT ownerEmail AS proposerEmail, ItemID AS proposedItemID
               FROM gameswapDB.Item
      ) AS M1
   WHERE M1.proposerEmail=(SELECT ownerEmail FROM Item WHERE itemID=@ItemID)
   UNION
   -- when I am the counterparty of the swap, select the proposer_rate
         SELECT counterpartyEmail AS Email, proposer_rate AS rate
   FROM gameswapDB.SwapRecord
      NATURAL JOIN(
         SELECT ownerEmail AS counterpartyEmail, ItemID AS desiredItemID
               FROM gameswapDB.Item
      ) AS M2
   WHERE M2.counterpartyEmail=(SELECT ownerEmail FROM Item WHERE
itemID=@ItemID)
) AS M) AS Rating
FROM Item
NATURAL JOIN
(SELECT first_name, last_name, email AS ownerEmail, postalCode FROM User) AS U
WHERE Item.itemID=@ItemID;
```

- Change the background color of Distance based on its value
- Join SwapRecord and Item table and calculate the number of unaccepted swaps (same as unaccepted swap subtask in main menu), store as **"Unaccepted swaps"**

```
SELECT COUNT(*)
FROM gameswapDB.SwapRecord NATURAL JOIN (
   SELECT ownerEmail AS counterpartyEmail, ItemID AS desiredItemID
   FROM gameswapDB.Item
) AS M
WHERE M.counterpartyEmail=@UserID AND SwapRecord.status IS NULL;
```

- Join SwapRecord with Item Table (join twice for desiredItem and proposedItem separately) and calculate the number of unrated swaps (same as the unrated swap subtask in main menu), store as **"unrated swaps"**,

```
SELECT COUNT(*)
FROM gameswapDB.SwapRecord NATURAL JOIN (
        SELECT ownerEmail AS proposerEmail, ItemID AS proposedItemID
        FROM gameswapDB.Item
)AS M1 NATURAL JOIN (
        SELECT ownerEmail AS counterpartyEmail, ItemID AS desiredItemID
        FROM gameswapDB.Item
)AS M2
WHERE ((proposerEmail=@UserID AND proposer_rate IS NULL) OR
(counterpartyEmail=@UserID AND counterparty_rate IS NULL)) AND (status=1);
```

- If number of unaccepted swaps<=5 and number of unrated swaps<=2
  - Show ***Propose Swap*** button, keep the item that is currently viewed as @DesiredItemID

Propose Swaps

Abstract Code

- Upon user clicked ***Propose Swap*** button
- Change the background color of Distance based on its value (distance is already calculated before)
- Query Item Table where (Item.ownerEmail==@UserID) and (item NOT IN pending or accepted swaps)

```
SELECT itemID, CASE
WHEN Item.itemID IN (SELECT itemID FROM BoardGame) THEN 'Board Game'
WHEN Item.itemID IN (SELECT itemID FROM CardGame) THEN 'Card Game'
WHEN Item.itemID IN (SELECT itemID FROM JigsawPuzzle) THEN 'Jigsaw Puzzle'
WHEN Item.itemID IN (SELECT itemID FROM ComputerGame) THEN 'Computer Game'
WHEN Item.itemID IN (SELECT itemID FROM VideoGame) THEN 'Video Game'
ELSE 'Error Type'
END AS Game_Type
, title, game_condition
FROM Item
WHERE ownerEmail = @UserID AND
                  (itemID NOT IN (SELECT proposedItemID
                FROM SwapRecord
                WHERE status = 1 or status IS NULL)) AND
                  (itemID NOT IN (SELECT desiredItemID
              FROM SwapRecord
              WHERE status = 1 or status IS NULL));
```

- Show items as radio check box
- When user click ***Confirm*** button
  - If check box is selected:
    - Query SwapRecord Table whether the same swaprecord has been submitted before

```
SELECT COUNT(*)
FROM SwapRecord
WHERE proposedItemID = @ProposedItemID AND desiredItemID = @DesiredItemID;
```

      - Return error message “You cannot send the same swap request that was rejected before again”
    - Else
      - Insert to SwapRecord Table, with status as NULL and propose_date as current time

```
INSERT INTO SwapRecord
(proposedItemID,desiredItemID,status,propose_date,decide_date,proposer_rate,counterparty_rate)
VALUES  (@ProposedItemID,@DesiredItemID,NULL,CURDATE(),NULL,NULL,NULL);
```

- Else:
  - Display error message “Please select an item to swap”

Accept/Reject Swaps

Abstract Code

- Upon user click ***Unaccepted swaps***
- Join User (calculate rating along the way with a procedure similar to my rating subtask in main menu) and Item Table as a temporary UserItem Table, join SwapRecord with UserItem (twice, as proposedItem and desiredItem separately). Select swap record where status IS NULL and counterpartyEmail (ownerEmail of the desiredItem)==@UserID, show ***Accept*** and ***Reject*** buttons

```
WITH UserItem AS (
SELECT email, postalCode, nick_name,
   (SELECT AVG(rate)
          FROM(
       SELECT proposerEmail as Email, counterparty_rate AS rate
       FROM SwapRecord NATURAL JOIN(
          SELECT ownerEmail AS proposerEmail, ItemID AS proposedItemID
                     FROM Item
                   ) AS M1
       WHERE M1.proposerEmail=email
       UNION
                  SELECT counterpartyEmail AS Email, proposer_rate AS rate
       FROM SwapRecord NATURAL JOIN(
          SELECT ownerEmail AS counterpartyEmail, ItemID AS desiredItemID
                FROM Item
       ) AS M2
       WHERE M2.counterpartyEmail=email
    ) AS M)
    AS rating, itemID, title
FROM   User
JOIN   Item
ON     User.email = Item.ownerEmail)
SELECT  SwapRecord.recordID, SwapRecord.propose_date AS Date, CUserItem.title AS
"Desired Item",
                  PUserItem.nick_name AS "Proposer", PUserItem.rating AS "Rating",
                  cal_dist(PUserItem.postalCode, CUserItem.postalCode) AS Distance,
                  PUserItem.title AS "Proposed Item"
FROM   SwapRecord
JOIN   UserItem AS CUserItem
ON     SwapRecord.desiredItemID = CUserItem.itemID
JOIN   UserItem AS PUserItem
ON     SwapRecord.proposedItemID = PUserItem.itemID
WHERE   SwapRecord.status IS NULL AND CUserItem.email=@UserID
ORDER BY SwapRecord.propose_date;
```

- Update status and decide_date

```
UPDATE  SwapRecord SET status = 1, decide_date = CURDATE() WHERE   recordID =
@RecordID;
```

- Return contact information for swap proposer

```
WITH UserItemPhone AS (
  SELECT  email, first_name, phoneNumber, phone_type, itemID
        FROM User
  JOIN Item
        ON User.email = Item.ownerEmail
        JOIN Phone
  ON User.email = Phone.ownerEmail)
SELECT  email, first_name, phonenumber, phone_type
FROM    UserItemPhone
JOIN    (Select proposedItemID
                FROM    SwapRecord
                WHERE   recordID = @RecordID) AS t1
ON      UserItemPhone.itemID = t1.proposedItemID;
```

- If user click ***Reject*** button
  - Update status and decide_date

```
UPDATE  SwapRecord SET status=0, decide_date= CURDATE() WHERE recordID = RecordID;
```

Swap History

Abstract Code

- Upon user click ***Swap history*** button
- Join User Table and Item Table as a temporary UserItem Table, Join with SwapRecord Table (twice, as proposedItem and desiredItem separately). Query SwapRecord where proposerEmail==@UserID or counterpartyEmail==UserID (i.e. ownerEmail of proposedItem or desired Item). Show rating box (the lower part of the table)

```
WITH
UserItem AS (
   SELECT  email, postalCode, first_name, last_name, nick_name, itemID, title, description,
game_condition
            FROM User JOIN Item ON User.email = Item.ownerEmail),
MySwapRecord AS (
   SELECT  recordID, PUserItem.email AS proposerEmail, CUserItem.email AS
counterpartyEmail, proposedItemID, desiredItemID, status, propose_date, decide_date,
proposer_rate, counterparty_rate, PUserItem.title AS Ptitle, CUserItem.title AS Ctitle,
PUserItem.nick_name AS Pnick_name, CUserItem.nick_name AS Cnick_name
            FROM SwapRecord
   JOIN UserItem AS PUserItem ON SwapRecord.proposedItemID = PUserItem.itemID
            JOIN UserItem AS CUserItem ON SwapRecord.desiredItemID = CUserItem.itemID
   WHERE   (PUserItem.email = @UserID OR CUserItem.email = @UserID) AND status IS NOT
NULL)
SELECT  propose_date AS "Proposed Date", decide_date AS "Accepted/Rejected Date", status
AS "Swap Status",
          CASE WHEN MySwapRecord.proposerEmail = @UserID THEN "Proposer"
     WHEN MySwapRecord.counterpartyEmail = @UserID THEN "Counterparty"
          END AS "My Role",
  Ptitle AS "Proposed Item", Ctitle AS "Desired Item",
  CASE WHEN MySwapRecord.proposerEmail != @UserID THEN Pnick_name
                     WHEN MySwapRecord.counterpartyEmail != @UserID THEN Cnick_name
  END AS "Other User",
  CASE WHEN (status = 1) AND MySwapRecord.proposerEmail = @UserID THEN
                            CASE WHEN proposer_rate IS NOT NULL THEN proposer_rate
                                       WHEN proposer_rate IS NULL THEN "$Rating"
     END
     WHEN (status = 1) AND MySwapRecord.counterpartyEmail = @UserID THEN
                            CASE WHEN counterparty_rate IS NOT NULL THEN counterparty_rate
        WHEN counterparty_rate IS NULL THEN "$Rating"
     END
  END AS "Rating"
FROM MySwapRecord ORDER BY decide_date DESC, propose_date;
```

- Sum number of swap record by whether @UserID is proposer or counterparty, by status and show as the upper part of the table

```
WITH
UserItem AS (
   SELECT email, postalCode, first_name, last_name, nick_name, itemID, title, description,
game_condition
            FROM User
   JOIN Item ON User.email = Item.ownerEmail),
MySwapRecord AS (
   SELECT recordID, PUserItem.email AS proposerEmail, CUserItem.email AS
counterpartyEmail, proposedItemID, desiredItemID, status, propose_date, decide_date,
proposer_rate, counterparty_rate, PUserItem.title AS Ptitle, CUserItem.title AS Ctitle,
PUserItem.nick_name AS Pnick_name, CUserItem.nick_name AS Cnick_name
            FROM SwapRecord
   JOIN UserItem AS PUserItem ON SwapRecord.proposedItemID = PUserItem.itemID
            JOIN UserItem AS CUserItem ON SwapRecord.desiredItemID = CUserItem.itemID
   WHERE (PUserItem.email = @UserID OR CUserItem.email = @UserID) AND status IS NOT
NULL)
SELECT  CASE WHEN proposerEmail = @UserID THEN "Proposer"
         WHEN counterpartyEmail = @UserID THEN "Counterparty"
      END AS "My role", COUNT(*) AS Total, sum(status) AS Accepted, sum(CASE WHEN
status = 0 THEN 1 ELSE 0 END) AS Rejected,
      sum(CASE WHEN status = 0 THEN 1 ELSE 0 END) / count(*) AS "Rejected %"
FROM MySwapRecord
GROUP BY CASE WHEN proposerEmail = @UserID THEN "Proposer"
          WHEN counterpartyEmail = @UserID THEN "Counterparty"
END;
```

- When user select rating for unrated swaps (save @RecordID from this operation)
  - If user is counterparty:

```
UPDATE  SwapRecord SET counterparty_rate = @Rating WHERE recordID = @RecordID;
```

  - If user is proposer

```
UPDATE  SwapRecord SET  proposer_rate = @Rating WHERE   recordID = @RecordID;
```

Rate Swap

Abstract Code

- Upon user click Unrated swaps link: Join User Table and Item Table as a temporary UserItem Table, join SwapRecord Table with UserItem Table (twice, as proposedItem and desiredItem separately). Select swaprecord where status==1 and ((proposerEmail==@UserID and proposer_rate IS NULL) or (counterpartyEmail==@UserID and counterparty_rate IS NULL)) (i.e. ownerEmail of proposedItem or desired Item). Show rating box

```
WITH UserItem AS (
   SELECT email, postalCode, first_name, last_name, nick_name, itemID, title, description,
game_condition FROM User
   JOIN Item ON User.email = Item.ownerEmail),
MySwapRecord AS (SELECT recordID, PUserItem.email AS proposerEmail, CUserItem.email
AS counterpartyEmail, proposedItemID, desiredItemID, status, propose_date, decide_date,
proposer_rate, counterparty_rate, PUserItem.title AS Ptitle, CUserItem.title AS Ctitle,
PUserItem.nick_name AS Pnick_name, CUserItem.nick_name as Cnick_name
   FROM SwapRecord
   JOIN UserItem AS PUserItem ON SwapRecord.proposedItemID = PUserItem.itemID
   JOIN UserItem AS CUserItem ON SwapRecord.desiredItemID = CUserItem.itemID
   WHERE   (PUserItem.email = @UserID OR CUserItem.email = @UserID) AND status IS NOT
NULL)
SELECT decide_date AS "Accepted/Rejected Date",
   CASE WHEN MySwapRecord.proposerEmail = @UserID THEN "Proposer"
      WHEN MySwapRecord.counterpartyEmail = @UserID THEN "Counterparty"
   END AS "My Role",
   Ptitle AS "Proposed Item", Ctitle AS "Desired Item",
   CASE WHEN MySwapRecord.proposerEmail != @UserID THEN Pnick_name
      WHEN MySwapRecord.counterpartyEmail != @UserID THEN Cnick_name
   END AS "Other User",
         CASE WHEN MySwapRecord.proposerEmail = @UserID AND
MySwapRecord.proposer_rate IS NULL THEN "$Rating"
      WHEN MySwapRecord.counterpartyEmail = @UserID AND
MySwapRecord.counterparty_rate IS NULL THEN "$Rating"
   END AS Rating
FROM   MySwapRecord
WHERE   status = 1 AND ((MySwapRecord.proposerEmail = @UserID AND
MySwapRecord.proposer_rate IS NULL) OR (MySwapRecord.counterpartyEmail = @UserID
AND MySwapRecord.counterparty_rate IS NULL))
ORDER BY decide_date DESC;
```

- When user select rating for unrated swaps (save @RecordID from this operation
  - If user is counterparty

```
UPDATE  SwapRecord SET counterparty_rate = @Rating WHERE recordID = @RecordID;
```

  - If user is proposer

```
UPDATE  SwapRecord SET  proposer_rate = @Rating WHERE   recordID = @RecordID;
```

  - Query unrated swaps again (same as the first step)
- If number of unrated swaps is 0: Return to **main** menu

Swap Details

Abstract Code

- Upon user click ***Detail*** button in swap history form (save @RecordID from this operation): Join User Table and item Table and Phone Table (Need to Left Join with Phone as the table on the right as not necessary everyone has a phone) as a temporary UserItemPhone table, query SwapRecord Table where recordID==@RecordID, join with UserItemPhone table (twice, as proposedItem and desiredItem separately). Query relevant information. [6]

[6] We query everything altogether and the MYSQL code in the next three pages should be executed all at once.

```
WITH TargetSwapRecord AS (
SELECT recordID, proposedItemID, desiredItemID, status, propose_date, decide_date,
proposer_rate, counterparty_rate
FROM  SwapRecord WHERE recordID = @RecordID),
UserItemPhone AS  (
SELECT  email, postalCode, first_name, last_name, nick_name, phoneNumber, phone_type,
share_phone, itemID, title, description, game_condition,
   CASE WHEN Item.itemID IN (SELECT itemID FROM BoardGame) THEN 'Board Game'
      WHEN Item.itemID IN (SELECT itemID FROM CardGame) THEN 'Card Game'
       WHEN Item.itemID IN (SELECT itemID FROM JigsawPuzzle) THEN 'Jigsaw Puzzle'
       WHEN Item.itemID IN (SELECT itemID FROM ComputerGame) THEN 'Computer Game'
       WHEN Item.itemID IN (SELECT itemID FROM VideoGame) THEN 'Video Game'
       ELSE 'Error Type'
   END AS game_Type
FROM User
JOIN Item ON User.email = Item.ownerEmail
LEFT JOIN Phone ON User.email = Phone.ownerEmail),
TargetSwapAllInfo AS (
SELECT recordID, PUserItemPhone.email AS proposerEmail, CUserItemPhone.email AS
counterpartyEmail, proposedItemID, desiredItemID, status, propose_date, decide_date,
proposer_rate, counterparty_rate, PUserItemPhone.title AS Ptitle, CUserItemPhone.title AS
Ctitle, PUserItemPhone.game_type AS Pgame_type, CUserItemPhone.game_type as
Cgame_type, PUserItemPhone.game_condition AS Pgame_condition,
CUserItemPhone.game_condition AS Cgame_condition, PUserItemPhone.description AS
Pdescription, PUserItemPhone.nick_name AS Pnick_name, CUserItemPhone.nick_name AS
Cnick_name, PUserItemPhone.first_name AS Pfirst_name, CUserItemPhone.first_name AS
Cfirst_name,PUserItemPhone.last_name AS Plast_name, CUserItemPhone.last_name AS
Clast_name, PUserItemPhone.postalCode AS PpostalCode, CUserItemPhone.postalCode AS
CpostalCode, PUserItemPhone.phoneNumber AS PphoneNumber,
CUserItemPhone.phoneNumber AS CphoneNumber, PUserItemPhone.phone_type AS
Pphone_type, CUserItemPhone.phone_type AS Cphone_type,
PUserItemPhone.share_phone AS Pshare_phone, CUserItemPhone.share_phone AS
Cshare_phone
     FROM TargetSwapRecord
     JOIN UserItemPhone AS PUserItemPhone
     ON TargetSwapRecord.proposedItemID = PUserItemPhone.itemID
     JOIN UserItemPhone AS CUserItemPhone
     ON TargetSwapRecord.desiredItemID = CUserItemPhone.itemID)
SELECT
```

```
-- swap details
propose_date AS Proposed, decide_date AS "Accepted/Rejected", status AS Status,
CASE WHEN TargetSwapAllInfo.proposerEmail = @UserID THEN "Proposer"
        WHEN TargetSwapAllInfo.counterpartyEmail = @UserID THEN "Counterparty"
END AS "My Role",
CASE WHEN TargetSwapAllInfo.proposerEmail = @UserID THEN
        CASE WHEN proposer_rate IS NOT NULL THEN proposer_rate
                ElSE "$Rating"
        END
        WHEN TargetSwapAllInfo.counterpartyEmail = @UserID THEN
        CASE WHEN counterparty_rate IS NOT NULL THEN counterparty_rate
                ElSE "$Rating"
        END
END AS "Rating left",
-- other user details
CASE WHEN proposerEmail =@UserID THEN Cnick_name
       WHEN counterpartyEmail =@UserID THEN Pnick_name
END AS "Nickname",
cal_dist(PpostalCode, CpostalCode) AS Distance,
CASE WHEN status = 1 THEN
       CASE WHEN proposerEmail = @UserID THEN Cfirst_name
              WHEN counterpartyEmail = @UserID THEN Pfirst_name
       END
       ELSE NULL
END AS "Name",
CASE WHEN status = 1 THEN
       CASE WHEN proposerEmail = @UserID THEN counterpartyEmail
              WHEN counterpartyEmail = @UserID THEN proposerEmail
       END
       ELSE NULL
END AS "Email",
CASE WHEN status = 1 THEN
       CASE WHEN (proposerEmail = @UserID AND Cshare_phone=1) THEN CphoneNumber
          WHEN (counterpartyEmail = @UserID AND Pshare_phone=1) THEN PphoneNumber
       ELSE NULL
       END
       ELSE NULL
END AS "Phone Number",
CASE WHEN status = 1 THEN
        CASE WHEN (proposerEmail = @UserID AND Cshare_phone=1) THEN Cphone_type
          WHEN (counterpartyEmail = @UserID AND pshare_phone=1) THEN Pphone_type
           ELSE NULL
       END
        ELSE NULL
```

```
END AS "Phone Type",
-- proposed item
proposedItemID AS "Item #", Ptitle AS "Title", Pgame_type AS "Game type",
Pgame_condition AS "Condition", Pdescription AS "Description",
-- desired item
desiredItemID AS "Item #", Ctitle AS "Title", Cgame_type AS "Game type", Cgame_condition
AS "Condition"
FROM TargetSwapAllInfo;
```

Update user info

Abstract Code

- Get number of unrated swaps and number of unapproved swaps ***Update my Info*** button only clickable when user has 0 unrated swaps, 0 unapproved swaps[7]

```
-- unrated swaps
SELECT COUNT(*)
FROM gameswapDB.SwapRecord NATURAL JOIN (
        SELECT ownerEmail AS proposerEmail, ItemID AS proposedItemID
        FROM gameswapDB.Item
)AS M1 NATURAL JOIN (
        SELECT ownerEmail AS counterpartyEmail, ItemID AS desiredItemID
        FROM gameswapDB.Item
)AS M2
WHERE ((proposerEmail=@UserID2 AND proposer_rate IS NULL) OR
(counterpartyEmail=@UserID2 AND counterparty_rate IS NULL)) AND (status=1);
```

```
-- unapproved swaps
SELECT COUNT(*)
FROM gameswapDB.SwapRecord NATURAL JOIN (
        SELECT ownerEmail AS proposerEmail, ItemID AS proposedItemID
        FROM gameswapDB.Item
)AS M1 NATURAL JOIN (
        SELECT ownerEmail AS counterpartyEmail, ItemID AS desiredItemID
        FROM gameswapDB.Item
)AS M2
WHERE ((proposerEmail=@UserID) OR (counterpartyEmail=@UserID)) AND (status IS
NULL);
```

- Upon user click ***Update my Info*** button: Query User where user=@UserID and show the **Update User Info** form (with *Email* not updatable)

```
SELECT email FROM gameswapDB.User WHERE User.email=@UserID;
```

- If data validation is successful for all fields (*Email, Nick_Name, Password, City, First_Name, State, Last_Name, Postal_Code, Phone_Number, Phone_Type, Share_Phone*), then
  - When ***Update*** button is clicked:
    - If @Postal_Code is not found in Location.Postal_Code:

```
SELECT COUNT(*) FROM gameswapDB.Location WHERE  ocation.postalCode=@Postal_Code;
```

      - Go back to **Update User Info** form, with error message “Please enter valid Postal Code”

[7] Following the instruction, unapproved swaps here include both the swaps where the current user is the proposer and the swaps where the current user is the counterparty

- Else if @Postal_Code is found in Location.postalCode and (@City!=Location.sity or State!=Location.state)

```
SELECT city,state FROM gameswapDB.Location WHERE Location.postalCode=@Postal_Code;
```

  - Go back to **Update User Info** form, with error message and suggestion for City and State info
- Else if @Phone is found in Phone and phone.ownerEmail!=@UserID:

```
SELECT COUNT(*) FROM gameswapDB.Phone WHERE Phone.phoneNumber=@Phone_Number
AND Phone.ownerEmail!=@UserID;
```

  - Go back to **Update User Info** form, with error message "This phone is used by someone else"
- Else
  - Update User and Phone Table

```
UPDATE gameswapDB.User SET
postalCode=@Postal_Code,password=@Password,first_name=@First_Name,last_name=@Last
_Name,nick_name=@Nick_Name WHERE email=@UserID;
```

```
UPDATE gameswapDB.Phone SET phoneNumber=@Phone_Number,
phone_type=@Phone_Type, share_phone=@Share_Phone
WHERE ownerEmail=@UserID;
```

  - Go to **Login** form