Table of Contents

Abstract Code with SQL

Enter Email Address

Abstract Code

- User enters *email* ('$Email') input field.
- If the data validation is successful for the *email* input field, then:
    - When ***Submit*** button is clicked, find if the household already exists in the database based on the email:

```
SELECT Email FROM Household WHERE Email= '$Email';
```

        - If the email already exists in the database for the email entered:
            - Show an error message and go back to the **Email Address Entry** form.
        - Else:
            - Save the value of '$Email' for inserting the household into database later.
            - Take the user to the **Postal Code Entry** form.
- Else, show an error message and go back to the **Email Address Entry** form.

Enter Postal Code

Abstract Code

- User enters 5-digit *postal_code* ('$PostalCode') input field.
- If the basic data type validation is successful at the front end for the *postal_code* ('$PostalCode') input field, then:
    - When ***Submit*** button is clicked, find if the postal code value entered can match any in the pre-defined postal code in PostalCodes table:

```
SELECT PostalCode, City, State, Longitude, Latitude FROM PostalCodes WHERE
PostalCode='$PostalCode';
```

        - If the postal code is not in the pre-defined PostalCodes table:
            - Show an error message and go back to the **Postal Code Entry** form.
        - Else:
            - Show user the corresponding city and state of the postal code entered, and ask the user to confirm:
                - If the user clicks the ***No*** button:
                    - Show an error message and go back to the **Postal Code Entry** form to re-entry.
                - Else:
                    - Store the '$PostalCode' value for further use when inserting a new record into the household table.
            - Take the user to the **Phone Number Entry** form.
- Else, show an error message and go back to the **Postal Code Entry** form.

Enter Phone Number

Abstract Code

- User selects if he/she wants to enter the phone number.
- If the user selects “Yes”, then:
    - Let the user enter the 3-digit *area_code* ('$AreaCode') input field and the 7-digit *phone_number* ('$RemainingSevenDigits') input field and select from the *phone_type* ('$PhoneType') dropdown menu.
        - Once the user clicks ***Next*** button, if the data validation is successful at the front end for the input fields, check if the phone number already exists in the database:

```
SELECT Email from Phone WHERE Area='$AreaCode' AND
RemainingSevenDigits='$RemainingSevenDigits';
```

            - If the phone number does not exist in the database:
                - Add the phone number to the Phone table together with the previously entered email.

```
INSERT INTO Phone (Area, RemainingSevenDigits, PhoneType, Email)
VALUES ('$AreaCode', '$RemainingSevenDigits', '$PhoneType', '$Email');
```

                - Take the user to the **Household Info** form
            - Else:
                - Display an error message and go back to the **Phone Number Entry** form.
        - Else:
            - Display an error message and go back to the **Phone Number Entry** form.
- Else, take the user to the **Household Info** form.

Enter Household Info

Abstract Code

- User selects from the *household_type* ('$HouseholdType') dropdown menu, enters the *square_footage* ('$SquareFootage'), chooses the *number_of_occupants* ('$NumberOfOccupants') and *number_of_bedrooms* ('$NumberOfBedrooms')
- If the data validation is successful for the input fields:
    - Once the user clicks ***Next*** button:
        - Save the entered household info to the Household table, together with the previously entered email and postal code.

```
INSERT INTO Household (Email, SquareFoot, Bedroom, Occupants, HouseholdType,
PostalCode) VALUES ('$Email', '$SquareFootage', '$NumberOfBedrooms',
'$NumberOfOccupants', '$HouseholdType', '$PostalCode');
```

        - Take the user to the **Bathroom Entry** form.
- Else:
    - Show an error message and go back to the **Household Info** form.

Add Bathroom

Abstract Code

- User selects *bathroom_type* ('$BathroomType'):
    - If the user selects half bathroom:
        - User selects/enters *number_of_sinks* ('$NumberOfSinks'), *number_of_commods* ('$NumberOfCommodes'), *number_of_bidets* ('$NumberOfBidets'), *name* ('$BathroomName').
    - If the user selects full bathroom, find if there already exists a primary full bathroom for the current household:

```
SELECT Email, [Order] from FullBathroom WHERE Email= '$Email' AND PrimaryBathroom= 'Yes';
```

        - If there already exists a primary full bathroom for the current household, the *primary_bathroom* ('$PrimaryBathroom') checkbox should be disabled and the values should be set to 'No' by default.
        - User selects *number_of_sinks* ('$NumberOfSinks'), *number_of_commods* ('$NumberOfCommodes'), *number_of_bidets* ('$NumberOfBidets'), *number_of_bathtubs* ('$NumberOfBathtubs'), *number_of_showers* ('$NumberOfShowers'), *number_of_tub_showers* ('$NumberOfTubShowers'), *primary_bathroom* ('$PrimaryBathroom') input fields.
- Once the user clicks ***Add***:
    - Check if the values entered are valid and satisfy the business logic constraints:
        - If valid:
            - Increase the system variable '$BathroomOrder' by 1 (It should be an int starting from 1).
            - If the user entered a full bathroom, save the info to FullBathroom:

```
INSERT INTO FullBathroom (Email, [Order], Sinks, Commodes, Bidet, Bathtub, Tub_Shower, Shower, PrimaryBathroom) VALUES ('$Email', '$BathroomOrder', '$NumberOfSinks', '$NumberOfCommodes', '$NumberOfBidets', '$NumberOfBathtubs', '$NumberOfTubShowers', '$NumberOfShowers', '$PrimaryBathroom');
```

            - If the user entered a half bathroom, save the info to HalfBathroom:

```
INSERT INTO HalfBathroom (Email, [Order], Sinks, Commodes, Bidet, Name) VALUES ('$Email', '$BathroomOrder', '$NumberOfSinks', '$NumberOfCommodes', '$NumberOfBidets', '$BathroomName');
```

            - Take the user to the **Bathroom Listing** form.
        - Else:
            - Show an error message and go back to the **Bathroom Entry** form.

Show Bathroom List

Abstract Code

- Query all bathrooms from the FullBathroom table and HalfBathroom table for the current household. Show a table that contains the bathroom's order, type, and if it is a primary one for all bathrooms.

```
SELECT [Order], 'Full' AS Type, PrimaryBathroom from FullBathroom WHERE FullBathroom.Email='$Email' UNION SELECT [Order], 'Half' AS Type, 'No' AS PrimaryBathroom from HalfBathroom WHERE HalfBathroom.Email='$Email' ORDER BY [Order];
```

- If the user clicks ***Add Another Bathroom*** button:
  - Return to the **Bathroom Entry** form.
- If the user clicks ***Next*** button.
  - Take the user to the **Appliance Entry** form.

Add Appliance

Abstract Code

- User selects the *appliance_type* ('$ApplianceType') dropdown menu:
- Depending on the type chosen, prompt the input fields for manufacturer name, model name, and the properties for that type. Query the dropdown list values for manufacturer names from the database:

```
SELECT Manufacturer FROM Manufacturer;
```

- User selects from *manufacturer* ('$Manufacturer') dropdown menu, optionally enters the *model_name* ('$ModelName').
- If the *appliance_type* ('$ApplianceType') is refrigerator/freezer:
  - User selects from the *refrigerator_freezer_type* ('$RefrigeratorFreezerType') dropdown menu.
- If the *appliance_type* ('$ApplianceType') is cooker:
  - User checks the *cooker_type_oven* ('$CookerTypeOven') and/or *cooker_type_cooktop* ('$CookerTypeCooktop') checkbox (allow multiple checks).
    - If *cooker_type_oven* ('$CookerTypeOven') is checked:
      - User selects the *oven_heat_source* ('$OvenHeatSource') checkbox (multiple values allowed).
      - User selects from the *oven_type* ('$OvenType') dropdown menu.
    - If *cooker_type_cooktop* ('$CookerTypeCooktop') is checked:
      - User selects from the *cooktop_heat_source* ('$CooktopHeatSource') dropdown menu.
    - If none is checked:
      - Show user an error message and go back to **Appliance Entry** form.
- If the *appliance_type* ('$ApplianceType') is washer:
  - User selects from the *loading_type* ('$LoadingType') dropdown menu.
- If the *appliance_type* ('$ApplianceType') is dryer:
  - User selects from the *dryer_heat_source* ('$DryerHeatSource') dropdown menu.
- If the *appliance_type* ('$ApplianceType') is TV:
  - User selects from the *TV_display_type* ('$TVDisplayType') dropdown menu.
  - User enters the *TV_display_size* ('$TVDisplaySize') input field.

- User selects from the *TV_maximum_resolution* ('$TVMaximumResolution') dropdown menu.

- Once the user clicks the ***Add*** button:
  - If the values entered are valid and satisfy the business logic constraints:
    - Increase the system variable '$ApplianceOrder' by 1 (It should be an int starting from 1).
    - Save the info entered to the corresponding table for each type:
      - If entered a refrigerator/freezer:

```
INSERT INTO RefrigeratorFreezer (Email, [Order], Manufacturer, ModelName, Type, SubType) VALUES ('$Email', '$ApplianceOrder', '$Manufacturer', '$ModelName', 'RefrigeratorFreezer', '$RefrigeratorFreezerType');
```

      - If entered a cooker:

```
INSERT INTO Cooker (Email, [Order], Manufacturer, ModelName, Type) VALUES ('$Email', '$ApplianceOrder', '$Manufacturer', '$ModelName', 'Cooker');
```

        - If *cooker_type_oven* ('$CookerTypeOven') is checked:

```
INSERT INTO Cooker_CookerType (Email, [Order], CookerType) VALUES ('$Email', '$ApplianceOrder', 'Oven');
```

```
INSERT INTO Oven (Email, [Order], OvenType) VALUES ('$Email', '$ApplianceOrder', '$OvenType');
```

```
INSERT INTO Oven_HeatSource (Email, [Order], HeatSource) VALUES ('$Email', '$ApplianceOrder', '$OvenHeatSource');
```

        - If *cooker_type_cooktop* ('$CookerTypeCooktop') is checked:

```
INSERT INTO Cooker_CookerType (Email, [Order], CookerType) VALUES ('$Email', '$ApplianceOrder', 'Cooktop');
```

```
INSERT INTO Cooktop (Email, [Order], HeatSource) VALUES ('$Email', '$ApplianceOrder', '$CooktopHeatSource');
```

      - If entered a washer:

```
INSERT INTO Washer (Email, [Order], Manufacturer, ModelName, Type, LoadingType) VALUES ('$Email', '$ApplianceOrder', '$Manufacturer', '$ModelName', 'Washer', '$LoadingType');
```

      - If entered a dryer:

```
INSERT INTO Dryer (Email, [Order], Manufacturer, ModelName, Type, HeatSource) VALUES ('$Email', '$ApplianceOrder', '$Manufacturer', '$ModelName', 'Dryer', '$DryerHeatSource');
```

      - If entered a TV:

```
INSERT INTO TV (Email, [Order], Manufacturer, ModelName, Type, DisplayType, MaxResolution, Size) VALUES ('$Email', '$ApplianceOrder', '$Manufacturer', '$ModelName', 'TV', '$TVDisplayType', '$TVMaximumResolution', '$TVDisplaySize');
```

    - Take the user to the **Appliance Listing** form.
  - Else:

- Show an error message and go back to the **Appliance Entry** form.

Show Appliance List

Abstract Code

- Query all appliances from the appliance-related tables for the current household. Show a table that contains the appliance's order, type, manufacturer, and model name for all appliances.

```
SELECT [Order], Type, Manufacturer, ModelName from RefrigeratorFreezer WHERE
RefrigeratorFreezer.Email='$Email'
UNION
SELECT [Order], Type, Manufacturer, ModelName from Cooker WHERE Cooker.Email='$Email'
UNION
SELECT [Order], Type, Manufacturer, ModelName from Washer WHERE
Washer.Email='$Email' UNION
SELECT [Order], Type, Manufacturer, ModelName from Dryer WHERE Dryer.Email='$Email'
UNION
SELECT [Order], Type, Manufacturer, ModelName from TV WHERE TV.Email='$Email'
ORDER BY [Order];
```

- If the user clicks ***Add Another Appliance*** button:
  - Return to the **Appliance Entry** form.
- If the user clicks ***Next*** button.
  - Take the user to the **Wrapping Up** page.

View Top 25 Popular Manufacturers

Abstract Code

- User clicked ***View Top 25 Popular Manufacturers*** button from **View reports/query data** page.
- Run the **View Top 25 Popular Manufacturers** task: query for information about Manufacturer and Order, the two columns consist of data from RefrigeratorFreezer, Washer, Dryer, TV and Cooker:
- Find and display the top 25 popular manufactures which have the most appliances from the database:

```
SELECT * INTO Popular_Manufacturer FROM (
SELECT r.Manufacturer, r."Order" FROM RefrigeratorFreezer r
UNION
SELECT w.Manufacturer, w."Order" FROM Washer w
UNION
SELECT d.Manufacturer, d."Order" FROM Dryer d
UNION
SELECT tv.Manufacturer, tv."Order" FROM TV tv
UNION
SELECT c.Manufacturer, c."Order" FROM Cooker c) a;
```

```
SELECT Manufacturer, COUNT("Order") AS TotalOrder FROM Popular_Manufacturer
GROUP BY Manufacturer
ORDER BY TotalOrder DESC
LIMIT 25;
```

- If the user would like to find more information about Manufacturer, there is a ***Manufacturer*** dropdown button, when user clicks on the manufacturer who needs, the system retrieves data for the user. After the Manufacturer ('$Manufacturer') option is submitted, find and display Type, count of those appliances of the types under the selected Manufacturer:

```
SELECT * INTO Total_Appliance FROM (
SELECT r.Manufacturer, r.Type, r."Order" FROM RefrigeratorFreezer r
UNION
SELECT w.Manufacturer, w.Type, w."Order" FROM Washer w
UNION
SELECT d.Manufacturer, d.Type, d."Order" FROM Dryer d
UNION
SELECT tv.Manufacturer, tv.Type, tv."Order" FROM TV tv
UNION
SELECT c.Manufacturer, c.Type, c."Order" FROM Cooker c) a;

SELECT Type, COUNT("Order") AS TotalOrder FROM Total_Appliance
WHERE Manufacturer = '$Manufacturer'
GROUP BY Type;
```

- After displaying the information successfully, if the user clicks ***Back*** button, return to the **View reports/query data** page.

Search Manufacturer/Model

Abstract Code

- User clicked ***Search Manufacturer/Model*** button from **View reports/query data** page.
- Run the **Search Manufacturer/Model** task: query for information about Manufacturer and ModelName:
- The user enters the part of string of a manufacturer name or model name ('$NameString').
- If the user clicks the ***Submit*** button, find and display matched Manufacturer and ModelName, both with ascending order:

```
SELECT * INTO Manufacturer_Model FROM (
SELECT r.Manufacturer, r.ModelName FROM RefrigeratorFreezer r
UNION
SELECT w.Manufacturer, w.ModelName FROM Washer w
```

```
UNION
SELECT d.Manufacturer, d.ModelName FROM Dryer d
UNION
SELECT tv.Manufacturer, tv.ModelName FROM TV tv
UNION
SELECT c.Manufacturer, c.ModelName FROM Cooker c) a;

SELECT Manufacturer, ModelName FROM Manufacturer_Model
WHERE Manufacturer LIKE '%$NameString%' OR ModelName LIKE '%$NameString%'
ORDER BY Manufacturer ASC, ModelName ASC;
```

- If can match any part of a manufacturer name or model name, the searching cell would be highlighted with a light green background to tell the user that there are Manufacturer or ModelName matched.
- After displaying the fetched information, if user clicks ***Back*** button, return to the **View reports/query data** page.

View Average TV Display Size by State

Abstract Code

- User clicked ***View Average TV Display Size by State*** button from **View reports/query data**
- Run the **View Average TV Display Size by State** task: query for information about PostalCodes and TV where average Size by each state will be displayed.
- Find and display the State and its average TV size:

```
SELECT State, ROUND(AVG(Size), 10) as Average_Size
FROM PostalCodes pc
JOIN Household h ON h.PostalCode = pc.PostalCode
JOIN TV ON TV.Email = h.Email
GROUP BY State
ORDER BY State ASC;
```

- If the user would like to find more information about PostalCodes and TV categorized by each State, there is a ***State*** dropdown button.
- When the user selects the state ('$State') and clicks the ***Submit*** button, find and display DisplayType, MaxResolution, average_Size of the selected state:

```
SELECT DisplayType, MaxResolution, ROUND(AVG(Size),10) as Average_Size
FROM PostalCodes pc
JOIN Household h ON h.PostalCode = pc.PostalCode
JOIN TV ON TV.Email = h.Email
WHERE State = '$State'
GROUP BY DisplayType, MaxResolution
```

```
ORDER BY Average_Size DESC;
```

- After displaying the fetched information, if user clicks ***Back*** button, return to the **View reports/query data** page.

View Extra Fridge/Freezer Report

Abstract Code

- User clicked on ***Extra Fridge/Freezer Report*** button from **View reports/query data**
- Run the **View Extra Fridge/Freezer Report** task
- Count household number from Refrigerator/freezer table using Refrigerator/freezer.Email where Email appears more than once. Display the count number.
    - SQL:

```
SELECT COUNT(DISTINCT Email) AS AllHouseholdMutiFridge
 FROM RefrigeratorFreezer
 WHERE Email IN (SELECT Email FROM RefrigeratorFreezer GROUP BY Email
HAVING COUNT(Email)>1);
```

- JOIN Refrigerator/freezer table and Household table ON email, then JOIN Postalcodes table ON Postalcode. GROUP Email by State for Email that appears more than once and COUNT that email per State. Display top 10 state and pertinent household count by household count descending.
- Count household whose email appears more than once and also:
- Refrigerator/freezer.Type is chest freezers;
- Refrigerator/freezer.Type is upright freezer;
- Refrigerator/freezer.Type is something else;
- Calculate and display the percentage of households with multiple fridge/freezers in that state
    - with chest freezers;
    - with upright freezer;
    - with something else;
    - SQL:

```
 SELECT TOP 10 p.State, COUNT(DISTINCT r.Email) AS
NumberOfHouseHoldsWithMultipleFridgeFreezers,
    CAST(100 * CAST(Count(distinct(case when r.Type = 'chest freezer' then
r.Email end)) as INT)/CAST(count(DISTINCT r.Email) as float) as INT) as
ChestPercentag,
    CAST(100 * CAST(Count(distinct(case when Type = 'upright freezer' then
r.Email end)) as INT)/CAST(count(DISTINCT r.Email) as float) as INT) as
UprightPercentage,
    CAST(100 * CAST(count(distinct(case when r.Type != 'upright freezer' and
r.Type != 'chest freezer' then r.Email end)) as INT)/CAST(count(DISTINCT
r.Email) as float) as INT) as OtherPercentage
```

```
FROM RefrigeratorFreezer r
LEFT JOIN Household h ON h.Email = r.Email
JOIN PostalCodes p ON p.PostalCode = h.PostalCode
WHERE r.Email IN (
      SELECT r.Email
          FROM RefrigeratorFreezer r GROUP BY r.Email HAVING
COUNT(r.Email)>1 )
GROUP BY p.State
ORDER BY COUNT(DISTINCT r.Email) DESC;
```

- If the user clicks ***Back*** button - Return to the **View reports/query data** page.

View Laundry Center Report

Abstract Code

- User clicked on ***View Laundry Center Report*** button from **View reports/query data** to view the most common washer type and dryer heat source report:
- The logic for determining the most common types: We join the table household and postalcodes by postcode, then join the table washer and dryer. Then we group the washer loading type and dryer heat source by different states. And pick the most common type of each state by ranking them. When two types have the same count, we display all of the types of same count.

```
SELECT state, loadingtype, heatsource
FROM
(
SELECT state, loadingtype, heatsource,
RANK() OVER(PARTITION BY state ORDER BY StateCountbyLoadingType DESC)
TopLoadingType,
RANK() OVER(PARTITION BY state ORDER BY StateCountbyHeatSource DESC)
TopHeatSource
FROM
(
  SELECT state, loadingtype, heatsource,
  COUNT(loadingtype) AS StateCountbyLoadingType,
  COUNT(heatsource) AS StateCountbyHeatSource
  FROM
   (SELECT p.state, w.loadingtype, d.heatsource
   FROM Household h
   LEFT JOIN PostalCodes p ON p.postalcode = h.postalcode
   RIGHT JOIN Washer w ON h.Email=w.email
   RIGHT JOIN Dryer d ON h.Email=d.email
   ) info
  GROUP BY state, loadingtype, heatsource
) SubGroup
) Ranks
```

```
WHERE TopLoadingType=1 AND TopHeatSource=1;
```

- Display the result on a table.
- If the user clicks the ***View Washer-only Report*** button from **View reports/query data** to view the household count (as an integer), per state, where a household has a washing machine, but does not have a dryer:

```
SELECT state, COUNT(Email) AS HouseholdCount
FROM
(SELECT h.Email, p.state,
COUNT(w.loadingtype) OVER(PARTITION BY h.Email) AS WasherCount,
COUNT(d.heatsource) OVER(PARTITION BY h.Email) AS DryerCount
FROM Household h
LEFT JOIN PostalCodes p ON p.postalcode = h.postalcode
LEFT JOIN Washer w ON h.Email=w.email
LEFT JOIN Dryer d ON h.Email=d.email
) MachineCount
WHERE WasherCount>0 AND DryerCount=0
GROUP BY state
ORDER BY HouseholdCount DESC
```

- Display the results on a table.
- If the user clicks ***Back*** button, return to the **View reports/query data** page.

View Bathroom Statistics

Abstract Code

- User clicked on ***Bathroom Statistics*** button from **View reports/query data**
- Run the **View Bathroom Statistics** task:
- UNION Full_Bathroom table and Half_Bathroom table using Email and Order; Count bathroom number per Household using Order under each Email. Display the minimum, average, and maximum count of all bathrooms per Email.
    - SQL:

```
SELECT MIN(ab.TotalBathroom) AS
Minimum_Count_of_Toal_Bathroom_Per_household
, CAST(AVG(ab.TotalBathroom) AS DECIMAL(10,1))AS
Average_Count_of_Total_Bathroom_Per_household
, MAX(ab.TotalBathroom) AS
Maximum_Count_of_Total_Bathroom_Per_household
FROM (
SELECT abe.Email, COUNT(abe.[Order]) AS TotalBathroom
FROM
(SELECT Fullbathroom.Email, Fullbathroom.[Order] FROM Fullbathroom
UNION ALL
```

```
SELECT Halfbathroom.Email, Halfbathroom.[Order] FROM Halfbathroom ) AS
abe
GROUP BY abe.Email ) AS ab;
```

- Count half bathroom number from Half_Bathroom table using Email. Display the minimum, average, and maximum count of all half bathrooms per Email.
    - SQL:

```
SELECT MIN(hb.HalfBathroomCount) AS
Minimum_Count_of_Half_Bathroom_Per_household
, CAST(AVG(hb.HalfBathroomCount) AS DECIMAL(10,1))AS
Average_Count_of_Half_Bathroom_Per_household
, MAX(hb.HalfBathroomCount) AS
Maximum_Count_of_Half_Bathroom_Per_household
FROM (SELECT HalfBathroom.Email, COUNT(HalfBathroom.Email) AS
HalfBathroomCount
FROM HalfBathroom
GROUP BY HalfBathroom.Email) hb;
```

- Count full bathroom number from Full_Bathroom table using Email. Display the minimum, average, and maximum count of full bathrooms per Email.
    - SQL:

```
SELECT MIN(fb.FullBathroomCount) AS
Minimum_Count_of_Full_Bathroom_Per_household
, CAST(AVG(fb.FullBathroomCount) AS DECIMAL(10,1))AS
Average_Count_of_Full_Bathroom_Per_household
, MAX(fb.FullBathroomCount) AS
Maximum_Count_of_Full_Bathroom_Per_household
FROM (SELECT FullBathroom.Email, COUNT(FullBathroom.Email) AS
FullBathroomCount
FROM FullBathroom
GROUP BY FullBathroom.Email) fb;
```

- UNION Email and Commodes from Full_Bathroom table and Half_Bathroom table; GROUP commodes number by Email; Count commodes number per Email. Display the minimum, average, and maximum count of Commodes per Email.
    - SQL:

```
SELECT MIN(ac.TotalCommodes) AS
Minimum_Count_of_Total_Commodes_Per_household
, CAST(AVG(ac.TotalCommodes) AS DECIMAL(10,1))AS
Average_Count_of_Total_Commodes_Per_household
, MAX(ac.TotalCommodes) AS
Maximum_Count_of_Total_Commodes_Per_household
FROM (
SELECT ace.Email, SUM(ace.Commodes) AS TotalCommodes
FROM
(SELECT Fullbathroom.Email, Fullbathroom.Commodes FROM Fullbathroom
UNION ALL
SELECT Halfbathroom.Email, Halfbathroom.Commodes FROM Halfbathroom)
AS ace
```

```
GROUP BY ace.Email) AS ac;
```

- UNION sinks number and Email from Full_Bathroom table and Half_Bathroom table; GROUP sinks number by Email; Count sinks number under same Email. Display the minimum, average, and maximum count of sinks per Email.
    - SQL:

```
SELECT MIN(asinks.Totalsinks) AS
Minimum_Count_of_Total_sinks_Per_household
, CAST(AVG(asinks.Totalsinks) AS DECIMAL(10,1))AS
Average_Count_of_Total_sinks_Per_household
, MAX(asinks.Totalsinks) AS Maximum_Count_of_Total_sinks_Per_household
FROM (
SELECT ase.Email, SUM(ase.sinks) AS Totalsinks
FROM
(SELECT Fullbathroom.Email, Fullbathroom.Sinks FROM Fullbathroom
UNION ALL
SELECT Halfbathroom.Email, Halfbathroom.Sinks FROM Halfbathroom) AS ase
GROUP BY ase.Email) AS asinks;
```

- UNION Bidet number and Email from Full_Bathroom table and Half_Bathroom; GROUP Bidet number by Email; Count Bidet number under same Email. Display the minimum, average, and maximum count of Bidet per Email.
    - SQL:

```
SELECT MIN(abidet.TotalBidet) AS
Minimum_Count_of_Total_Bidet_Per_household
, CAST(AVG(abidet.TotalBidet) AS DECIMAL(10,1))AS
Average_Count_of_Total_Bidet_Per_household
, MAX(abidet.TotalBidet) AS Maximum_Count_of_Total_Bidet_Per_household
FROM (
SELECT abi.Email, SUM(abi.Bidet) AS TotalBidet
FROM
(SELECT Fullbathroom.Email, Fullbathroom.Bidet FROM Fullbathroom
UNION ALL
SELECT Halfbathroom.Email, Halfbathroom.Bidet FROM Halfbathroom) AS abi
GROUP BY abi.Email) AS abidet;
```

- Count Bathtub number from Full_Bathroom table using Full_Bathroom.Bathtub under same Email. Display the minimum, average, and maximum count of Bathtub per Email.
    - SQL:

```
SELECT MIN(abathtub.TotalBathtub) AS
Minimum_Count_of_Bathtub_Per_household
, CAST(AVG(abathtub.TotalBathtub) AS DECIMAL(10,1))AS
Average_Count_of_Bathtub_Per_household
, MAX(abathtub.TotalBathtub) AS Maximum_Count_of_Bathtub_Per_household
FROM (SELECT FullBathroom.Email, COUNT(FullBathroom.Bathtub) AS
TotalBathtub
FROM FullBathroom
GROUP BY FullBathroom.Email) abathtub;
```

- Count Shower number from Full_Bathroom table using Full_Bathroom.Shower under same Email. Display the minimum, average, and maximum count of Shower per Email.
    - SQL:

```
SELECT MIN(ashower.TotalShower) AS
Minimum_Count_of_Shower_Per_household
, CAST(AVG(ashower.TotalShower) AS DECIMAL(10,1))AS
Average_Count_of_Shower_Per_household
, MAX(ashower.TotalShower) AS Maximum_Count_of_Shower_Per_household
FROM (SELECT FullBathroom.Email, COUNT(FullBathroom.Shower) AS
TotalShower
FROM FullBathroom
GROUP BY FullBathroom.Email) ashower;
```

- Count Tub/Shower number from Full_Bathroom table using Full_Bathroom.Tub/Shower under same Email. Display the minimum, average, and maximum count of Tub/Shower per Email.
    - SQL:

```
SELECT MIN(atubshower.TotalTub_Shower) AS
Minimum_Count_of_Tub_Shower_Per_household
, CAST(AVG(atubshower.TotalTub_Shower) AS DECIMAL(10,1))AS
Average_Count_of_Tub_Shower_Per_household
, MAX(atubshower.TotalTub_Shower) AS
Maximum_Count_of_Tub_Shower_Per_household
FROM (SELECT FullBathroom.Email, COUNT(FullBathroom.Tub_Shower) AS
TotalTub_Shower
FROM FullBathroom
GROUP BY FullBathroom.Email) AS atubshower;
```

- UNION Email and Bidet number from Full_Bathroom table and Half_Bathroom table, then JOIN Household table ON Email, then JOIN Postalcodes table ON Postalcode; Group Bidet count number by State; Display the State with the most Bidet count and the count of Bidet in that state.
    - SQL:

```
SELECT SBidet.State, SBidet.StateBidet
FROM
 (SELECT State, SUM(Bidet) AS StateBidet, RANK() OVER(ORDER BY
SUM(Bidet) DESC) StateRank
  FROM (SELECT Fullbathroom.Email, Fullbathroom.Bidet FROM Fullbathroom
      UNION ALL SELECT Halfbathroom.Email, Halfbathroom.Bidet FROM
Halfbathroom
 ) AS abi
  LEFT JOIN Household ON Household.Email = abi.Email
  LEFT JOIN PostalCodes ON Postalcodes.Postalcode =Household.Postalcode
  GROUP BY State
) AS SBidet
WHERE SBidet.StateRank = 1;
```

- UNION Email and Bidet number from Full_Bathroom table and Half_Bathroom, then JOIN Household table ON Email, then JOIN Postalcodes table ON Postalcode; Group

Bidet count number by PostalCode; Display the postal code with the most Bidet count and the count of Bidet under that postal code.

- SQL:

```
SELECT SBidet.PostalCode, SBidet.PCBidet
FROM
 (SELECT Household.PostalCode, SUM(Bidet) AS PCBidet, RANK()
OVER(ORDER BY SUM(Bidet) DESC) PCRank
  FROM (SELECT Fullbathroom.Email, Fullbathroom.Bidet FROM Fullbathroom
      UNION ALL SELECT Halfbathroom.Email, Halfbathroom.Bidet FROM
Halfbathroom
 ) AS abi
     LEFT JOIN Household ON Household.Email = abi.Email
     LEFT JOIN PostalCodes ON Postalcodes.Postalcode
=Household.Postalcode
  GROUP BY Household.PostalCode
) AS SBidet
WHERE SBidet.PCRank = 1;
```

- Count household number from Full_Bathroom table using Full_Bathroom.Email, where PrimaryBathroom is YES, and Email only appears once in Full_Bathroom table and Half_Bathroom table. Display the household count number that meets the requirements.
  - SQL:

```
 SELECT COUNT(FullBathroom.Email) AS Household_SinglePriBathroom
FROM Fullbathroom
WHERE FullBathroom.PrimaryBathroom = 'Yes' AND FullBathroom.email
IN (
SELECT tbath.Email
FROM
(SELECT abe.Email, COUNT(abe.[Order]) AS TotalBathroom
FROM
(SELECT Fullbathroom.Email, Fullbathroom.[Order] FROM Fullbathroom
UNION ALL
SELECT Halfbathroom.Email, Halfbathroom.[Order] FROM Halfbathroom ) AS
abe
GROUP BY abe.Email) AS tbath
WHERE tbath.TotalBathroom = 1);
```

- Display the fetched results using tables.
- If the user clicks ***Back*** button - Return to the **View reports/query data** page.

Search/View Household Averages By Radius

Abstract Code

- User enters 5-digit *postal_code* ('$PostalCode) input field.

- If the basic data type validation is successful for the *postal_code* ('$PostalCode') input field and available dropdown menu for search radius choice ('$Radius') is selected, then:
    - When ***Submit*** button is clicked:
        - If the postal code value entered does not match any in the pre-defined postal code list:
            - Show an error message and go back to the **Postal Code Entry** form.
        - Else:
            - Query the households which are in the requested radius from households and address table by filtering out any postal code not in the search radius.
            - After we get selected households, by data return from query selected households, we calculate the average bathroom count, the average bedroom count, the average occupant count.
            - By querying the Bathroom table on selected households, we calculate the ratio of commodes to occupants.
            - By querying the Cooker, Refrigerator/freezer, Washer, Dryer, TV tables on selected households, we calculate the average number of appliances, and the most common heat source.

SQL:

```
with Center as (
select Latitude as CenterLatitude, Longitude as CenterLongitude from [dbo].[Posta
lCodes] where PostalCode = '$PostalCode'
),
AllPostalCodes as(
select PostalCode, (3959 * acos(cos( radians(CenterLatitude) ) * cos( radians( La
titude ) ) * cos( radians(CenterLongitude) - radians(Longitude) ) + sin( radians(
CenterLatitude) ) * sin( radians(Latitude)))) as Radius
from PostalCodes
cross join Center
),
ValidPostalCodes as (
select h.Email, AllPostalCodes.PostalCode, '$Radius' as Radius, h.Bedroom, h.Occupa
nts
from HouseHold h, AllPostalCodes
where AllPostalCodes.Radius <= '$Radius' and h.PostalCode = AllPostalCodes.PostalCo
de
),
BathInfo as (
select Email, count(*) as BathRoomCount, sum(Commodes) as CommodesCount
from (
Select Email, Commodes
from FullBathroom f
Where f.Email in (Select Email from ValidPostalCodes)
union
```

```
select Email, Commodes
from HalfBathroom h
Where h.Email in (Select Email from ValidPostalCodes)) as b
group by Email
),
Appliance as (
Select Email, count(Email) as c
from(
Select Email
from RefrigeratorFreezer r
Where r.Email in (Select Email from ValidPostalCodes)
union
Select Email
from Cooker c
Where c.Email in (Select Email from ValidPostalCodes)
union
select Email
from Washer w
Where w.Email in (Select Email from ValidPostalCodes)
union
select Email
from Dryer d
Where d.Email in (Select Email from ValidPostalCodes)
union
select Email
from TV t
Where t.Email in (Select Email from ValidPostalCodes)) as a
group by Email
),
Calculation as (
select ValidPostalCodes.PostalCode, ValidPostalCodes.Radius,
    CAST(CAST(sum(BathInfo.BathRoomCount) as float(10))/CAST(count(ValidPostalCod
es.Email) as float(10)) as decimal(5,1)) as AVG_BathRoom,
    CAST(CAST(sum(ValidPostalCodes.Bedroom) as float(10))/CAST(count(ValidPostalC
odes.Email) as float(10)) as decimal(5,1)) as AVG_Bedroom,
    CAST(CAST(sum(ValidPostalCodes.Occupants) as float(10))/CAST(count(ValidPosta
lCodes.Email) as float(10)) as int) as AVG_Occupants,
    CAST(CAST(sum(BathInfo.CommodesCount) as float(10))/CAST(sum(ValidPostalCodes
.Occupants) as float(10)) as decimal(5,2)) as CommadeToOccupantRatio,
    CAST(CAST(sum(Appliance.c) as float(10))/CAST(count(ValidPostalCodes.Email) a
s float(10)) as decimal(5,1)) as AVG_Appliance
from ValidPostalCodes, BathInfo, Appliance
where ValidPostalCodes.Email = BathInfo.Email and ValidPostalCodes.Email = Applia
nce.Email
```

```
group by ValidPostalCodes.PostalCode, ValidPostalCodes.Radius
),
HeatSource as (
Select Top 1 HeatSource as MostCommonHeatSource from(
select HeatSource
from Oven_HeatSource o
Where o.Email in (Select Email from ValidPostalCodes)
union
select HeatSource
from Dryer d
Where d.Email in (Select Email from ValidPostalCodes)
union
select HeatSource
from Cooktop c
Where c.Email in (Select Email from ValidPostalCodes)) as h
group by HeatSource
order by count(*) desc
)
select Calculation.*, HeatSource.MostCommonHeatSource from Calculation, HeatSourc
e;
```

- Else, show an error message and go back to the **Household Averages By Radius** form.