Table of Contents

1. **Welcome / View Report**
    - Show **'*Enter my household info*',** '***View reports / query date***' tabs.
    - Case tab clicked of:
        - Click **'*Enter my household info*', go 'Enter household Information'** Task
        - Click '***View reports / query date***', go to 'View reports' page
            - Do nothing until one of several buttons are pushed
            - Case button pushed of:
                - '***Top 25 Manufacturers***', go to 'Top 25 Manufacturers' page
                - ***'Manufacturer/model search***', go to 'Manufacturer/model search' page
                - '***Average TV display size by state***', go to 'Average TV display size by state' page
                - '***Extra fridge/freezer report***', go to 'Extra fridge/freezer report' page
                - '***Laundry center report***', go to 'Laundry center report' page
                - '***Bathroom statistics***', go to 'Bathroom statistics' page
                - '***Household averages by radius***', go to 'Household averages by radius' page

2. **Enter Household Information**

   Enter Email Address

   While no button is clicked, do nothing

    - Users enter *email* ('$email') and click the ***Next*** button, do:
        - Check for email format validity: <username>@<domain> in the front-end:
            - If email format validation is successful, then:
                - Query the database if the input *email* ('$email') address already exists:

                ```
                SELECT email FROM HouseHold WHERE HouseHold.email =
                '$email';
                ```

                - If the email exists in the database (HouseHold.email = '$email'), display the message that reads 'The email has already been used, please use another email.', and reset the input field.
                - Else, save the email ('$email') to the front-end state. then move to the next subtask, **Enter Postal Code.** (Do not write the email into the database at the stage)
            - Else: display the message that reads 'Invalid Email' and reset the input field.

Enter Postal Code

- While no button is clicked, do nothing.
- Users enter postal code ('$postal_code') in input fields and click on the ***Next*** button, do:
    - Query the database to verify if the postal code exists:

```
SELECT postal_code, city, state FROM PostalCode WHERE
PostalCode.postal_code = '$postal_code';
```

        - If the postal code exists (PostalCode.postal_code = '$postal_code'): display the confirmation of the postal code with matching city and state ('$postal_code', '$city', '$state'). Ask the user to confirm by clicking on the buttons:
            - If the ***Yes*** button is clicked: Save the postal code ('$postal_code') into the front-end state, then move on to the next subtask: '**Enter Phone Number'.**
            - Else: restart the **'Enter Postal Code'** task.
        - Else: display the message 'Invalid postal code' and reset the input fields.

Enter Phone Number

- If Yes/No toggle is switched to No, gray out input fields and only allow users to click ***Next*** button to ***'*Enter Household Information'** task.
- If Yes/No toggle is switched to Yes: Display the phone number input field.
- Users will be required to enter the area code ('$area_code'), and the 7-digit number ('$number'), as well as choose the phone type ('$phone_type') from a drop-down.
    - When the user hits the ***Next*** button, validate whether the area code ('$area_code'), number are in their right format (3 digits exact for area code and 7 digits exact for number field) on the front-end and then
        - If the format is incorrect, then display the error message: "Invalid phone number" and then reset the form
        - Else the format is correct, then query the database to verify if the input phone number exists.

```
SELECT area_code, number FROM PhoneNumber WHERE
PhoneNumber.area_code = '$area_code' AND PhoneNumber.number =
'$number';
```

        - If the phone number is found (PhoneNumber.area_code = '$area_code' and PhoneNumber.number = '$number'), display the message 'Phone number has already been used' and reset the input fields.
        - Else: Save the phone number into the front-end state and allow users to click ***Next*** to **'Enter Household Information'** task.

Enter Household Information

- ***Next*** button is clickable only when all inputs fields are filled.
- When the user fills in the input fields and clicks the ***Next*** button, do:
    - Save the household information into front-end state.
    - Write the email ('$email') and household information ('$home_type', '$square_footage', '$occupant', '$bedroom') and postal code ('$postal_code') into HouseHold table.

      ```
      INSERT INTO HouseHold (email, square_footage, occupant, bedroom,
      home_type, FK_HouseHold_postal_code_PostalCode_postal_code)
      VALUES ('$email', '$square_footage', '$occupant', '$bedroom',
      '$home_type', '$postal_code');
      ```
    - Write phone number ('$area_code', '$number', '$phone_type') into PhoneNumber table.

      ```
      INSERT INTO PhoneNumber (area_code, number, phone_type,
      FK_PhoneNumber_email_HouseHold_email)
      VALUES ('$area_code', '$number', '$phone_type', '$email');
      ```
    - Then move to the next task: **'Enter Bathroom Info'**.

3. Add Bathroom Details

Enter Bathroom Info

- User to choose from the tabs for the entry bathroom type ***Half*** or ***Full***.
- If user clicks the ***Half*** tab:
    - Render the **Half Bathroom** form.
    - The input information is flagged to insert to Half Bathroom table.
- If user clicks the ***Full*** tab:
    - Render **Full bathroom** form.
    - The input information is flagged to insert to Full Bathroom table.
    - If the house already has a primary full bathroom, the option to set this bathroom as primary will be disabled.
- User to input the details of each bathroom's facilities.
    - Half Bathroom: optional *name of the bathroom*('$name'), the quantities of *sink*('$sink'), *commode*('$commode') and *bidet* ('$bidet').
    - Full Bathroom: is_primary('$is_primary'), the quantities of *sink*('$sink'), *commode*('$commode'), *bidet* ('$bidet'), *bathtub*('$bathtub'), *shower* ('$shower') and *tub_shower*('$tub_shower').
- When the ***Add*** button is clicked:
    - Check the following conditions: A half bathroom has at least one sink, commode, and/or bidet; A full bathroom has at least one bathtub, shower, or tub/shower.
        - If the condition is not satisfied, Render an error message.
        - Else:
            - Add an order number ('$number') to each bathroom by the order in which it is entered.
            - Save all the information into the front-end state and move on to the subtask **'List Bathrooms Info'**.

List Bathrooms Info

- This bathroom list shows in tabular form the bathroom spec that has been added by the previous forms. The information shown is the bathroom order number ('$number'), the type of bathroom that they are flagged as full or half, and primary status ('$is_primary').
- While no button is clicked, do nothing.
- If users click the ***Add Another Bathroom***, go back to the **'Entry Bathroom Info'** task.
- If the ***Next*** button is clicked:
    - Write the email ('$email'), name of the bathroom('$name'), an order number ('$number'), and the quantities of sink('$sink'), commode('$commode') and bidet ('$bidet') into Half Bathroom table.

```
INSERT INTO Half (number, sink, commode, bidet, name,
FK_Half_email_HouseHold_email)
VALUES ('$number', '$sink', '$commode', '$bidet', '$name', '$email');
```

    - Write the email ('$email'), is_primary('$is_primary'), an order number ('$number'), and the quantities of *sink*('$sink'), *commode*('$commode'), *bidet* ('$bidet'), *bathtub*('$bathtub'), *shower* ('$shower') and *tub_shower*('$tub_shower') into Full Bathroom table.

```
INSERT INTO Full (number, sink, commode, bidet, is_primary, bathtub,
shower, tub_shower, FK_Full_email_HouseHold_email)
VALUES ('$number', '$sink', '$commode', '$bidet', '$is_primary', '$bathtub',
'$shower', '$tub_shower', '$email');
```

    - Then move to the next task: **Enter Appliance**

4. Add Appliance Details
Enter Appliance Info

- Users choose an appliance type from the 'appliance type' dropdown selection:
    - If the appliance type is a Cooker:
        - Render the **Cooker** form.
        - Prompt the appropriate manufacturers ('$manufactures_name') drop selection for user to choose and display *model name* ('$model_name') input.
        - The input information is flagged to insert to Cooker table.
        - Render the cooker spec options:
            - Cooker type: oven and cooktop; users are allowed to select both options.
            - If select Oven:
                - the input information is flagged to insert in Oven table.
                - Check box of Heat source: gas ('$gas'), electric ('$gas') and/or microwave ('$microwave'). Users are allowed to select more than one source.
                - Oven type ('$oven_type') dropdown selection: either convection, or conventional.

- If select Cooktop:
  - the input information is flagged to insert in Cooktop table.
  - Heat_source ('$heat_source') dropdown selection: gas, electric, radiant electric, or conventional.

- If the appliance type is a TV:
  - Render the **TV** form.
  - Prompt the appropriate manufacturers ('$manufactures_name') drop selection for user to choose and display *model name* ('$model_name') input.
  - Render the TV spec options:
    - Display Type ('$display_type') dropdown selection: tube, DLP, plasma, LCD, or LED
    - Display size ('$display_size') input field: input in fractional inches
    - Maximum resolution ('$maximun_resolution') dropdown selection:480i, 576i, 720p, 1080i, 1080p, 1440p, 2160p(4K), or 4320p (8K)
- If the appliance type is a Washer:
  - Render the **Washer** form.
  - Prompt the appropriate manufacturers ('$manufactures_name') drop selection for user to choose and display *model name* ('$model_name') input.
  - Render the washer spec options:
    - Loading type ('$loading_type') dropdown selection: either top or front.
- If the appliance type is a Dryer:
  - Render the **Dryer** form.
  - Prompt the appropriate manufacturers ('$manufactures_name') drop selection for user to choose and display *model name* ('$model_name') input.
  - Render the dryer spec options:
    - Heat source ('$heat_source') dropdown selection: gas, electric, or none (For dryers which do not utilize heat, such as a condensing dryer)
- If the appliance type is a Freezer(/refrigerator):
  - Render the **Freezer** form.
  - Prompt the appropriate manufacturers ('$manufactures_name') drop selection for user to choose and display *model name* ('$model_name') input.
  - Render the dryer spec options:
    - Model type ('$model_type') dropdown selection: Bottom freezer/refrigerator, French door refrigerator, side-by-side refrigerator, top freezer refrigerator, chest freezer, upright freezer

- When the ***Add*** button is clicked:
  - Add an order number to each appliance by the order in which it is entered.
  - write the information into the front-end state and move to the '**List Appliances Info'** subtask

Notice 1: In this stage, each appliance will have their own specific specs. For an example, "The type of refrigerator/freezer: Bottom freezer refrigerator, French door refrigerator, side-by-side refrigerator, top freezer refrigerator, chest freezer, or upright freezer."

Notice 2: The set of manufacturers is from a finite list, which can be updated by the development team.

List Appliances Info

- This appliance listing shows in tabular form the appliances that have been added by the previous form. The information shown is the appliance order number, the types of appliances that they are flagged as Cooker(Oven and/or Cooketop), TV, Washer, Dryer, Freezer (/Refrigeration), manufacturers ('$manufactures_name'), and models ('$model_name')
- While no button is clicked, do nothing.
- If "***Add another appliance"*** is clicked: go back to '**Enter Appliance Info'** task.
- If the ***Next*** button is pushed:
    - Write the appliance information saved in the front-end state into Database tables that they are flagged.
    - Cooker, TV, Washer, Dryer, Freezer (/Refrigeration) is set with auto-increment IDs and middleware will fetch cooker_id to associated Oven and/or Cooktop.
        - Cooker

```
INSERT INTO Cooker (FK_Cooker_email_HouseHold_email,
model_name, name)
VALUES ('$email', '$model_name', '$manufacturer_name');
```

            - If Cooker type is Oven

```
INSERT INTO Oven (FK_Oven_id_Cooker_cooker_id,
FK_Cooker_email_HouseHold_email, has_gas_heat_source,
has_electric_heat_source, has_microwave_heat_source, oven_type)
VALUES ('$cooker_id', '$email', '$has_gas', '$has_electric',
'$has_microwave', '$oven_type');
```

            - If Cooker type is Cooktop

```
INSERT INTO Cooktop (FK_Cooktop_id_Cooker_cooker_id,
FK_Cooker_email_HouseHold_email, heat_source)
VALUES ('$cooker_id', '$email', $heat_source');
```

        - TV

```
INSERT INTO TV (FK_tv_email_HouseHold_email, name,
model_name, display_type, display_size, maximum_resolution)
VALUES ('$email', '$manufacturer_name', '$model_name',
'$display_type', '$display_size', '$maximun_resolution');
```

        - Washer

```
INSERT INTO Washer (FK_Washer_email_HouseHold_email, name,
model_name, loading_type)
VALUES ('$email', '$manufacturer_name', '$model_name',
'$loading_type');
```

        - Dryer

```
INSERT INTO Dryer (FK_Dryer_email_HouseHold_email, name,
model_name, loading_type)
VALUES ('$email', '$manufacturer_name', '$model_name',
'$heat_source');
```

- Freezer

```
INSERT INTO Freezer (FK_Freezer_email_HouseHold_email, name,
model_name, model_type)
VALUES ('$email', '$manufacturer_name', '$model_name',
'$model_type');
```

- Next, render the '**Submission Complete'** page.
  - **'Submission Complete**' will provide option to go back to main page **'Welcome / View Report'.**

5. Top 25 Popular Appliance Report

- When users click the 'Top 25 Manufacturer Report'
  - Query the database to Display the 'Top 25 manufacturers'
  - Find manufacture in the Union of all types of appliances (cooker, Dryer, Freezer, TV, Washer), and group and count the same manufacture name. Display the top 25 in the descending count order.

```
SELECT name, COUNT (*) AS frequency FROM
(SELECT cooker_id, FK_cooker_email_HouseHold_email, model_name, name FROM
Cooker UNION
SELECT dryer_id, FK_Dryer_email_HouseHold_email, Model_name, name FROM Dryer
UNION
SELECT freezer_id, FK_Freezer_email_HouseHold_email, model_name, name FROM
Freezer UNION
SELECT tv_id, FK_tv_email_HouseHold_email, model_name, name FROM TV UNION
SELECT washer_id, FK_Washer_email_HouseHold_email, model_name, name FROM
Washer) Appliances
GROUP BY name
ORDER BY frequency DESC
LIMIT 25;
```

  - For each manufacturer of the 'top 25', there will be a drilldown report provided to show the appliance type and the count of it.
  - Distinguish each appliance by given an alias name as same as their type, find and count each appliance with the same specified manufacture name (in the Union of appliance).

```
SELECT appliance_type, count(*) AS frequency FROM
(SELECT 'Cooker' AS appliance_type, cooker_id, FK_cooker_email_HouseHold_email,
model_name, name FROM Cooker UNION
SELECT 'Dryer' AS appliance_type, dryer_id, FK_Dryer_email_HouseHold_email,
Model_name, name FROM Dryer UNION
SELECT 'Freezer' AS appliance_type, freezer_id, FK_Freezer_email_HouseHold_email,
model_name, name FROM Freezer UNION
SELECT 'TV' AS appliance_type, tv_id, FK_tv_email_HouseHold_email, model_name, name
from TV UNION
```

```
SELECT 'Washer' AS appliance_type, washer_id, FK_Washer_email_HouseHold_email,
model_name, name FROM Washer) Appliances
WHERE name = '$manufacturer_name'
GROUP BY name, appliance_type;
```

6. **Manufacturer / Model Search Report**
   - When users click 'Manufacturer/ Model Search', collect user input and store it in variable ('$part_manufacture/model')
   - Collect information from Cooker
   - Collect information from Dryer
   - Collect information from Freezer
   - Collect information from TV
   - Collect information from Washer
   - Union the tables for Cooker, Dryer, Freezer, TV, and Washer and store in an intermediate table
   - Query the intermediate table for distinct model_name and manufacturer name from the intermediate table with the LIKE operator on the user input for the model_name in the where clause
   - Group by model_name and manufacturer name in ascending order

```
SELECT distinct(model_name), name FROM
(SELECT cooker_id, FK_cooker_email_HouseHold_email, model_name, name FROM
Cooker UNION
SELECT  dryer_id, FK_Dryer_email_HouseHold_email, Model_name, name FROM Dryer
UNION
SELECT  freezer_id, FK_Freezer_email_HouseHold_email, model_name, name FROM
Freezer UNION
SELECT tv_id, FK_tv_email_HouseHold_email, model_name, name from TV UNION
SELECT  washer_id,  FK_Washer_email_HouseHold_email, model_name, name FROM
Washer ) Appliances
WHERE model_name LIKE "%$part_manufacture/model%"
ORDER BY model_name, name ASC
```

7. **Average TV Display Size by State Report**
   - Collect information from PostalCode table and join the information from the Household table using common postal code attribute, then join the information from the TV table using the common email attribute for the Household table and TV table.
   - Select state, postal_code, email, display_size columns from the above intermediate results.
   - Compute the averages for the display_size for the states, format as required
   - Group the results by state and display in ascending order

```
SELECT state, FORMAT(AVG(display_size), '2.#') AS Average_size FROM
(select state, postal_code, email, display_size from postalCode
join Household
ON postal_code = Household.FK_HouseHold_postal_code_PostalCode_postal_code
```

```
join TV
ON Household.email = TV.FK_tv_email_HouseHold_email) ALL_TV_IN_STATE
group by state
ORDER BY STATE ASC
```

Drill down report by state

- Provide a mechanism for the user to choose a state as input and store in a variable '$State'
- Query the intermediate table from above for screen type and maximum resolution
- Join two tables
- Add condition clause for the user input state to show result for that state

```
WITH x1 AS (
SELECT state, AVG(display_size) AS Average_size FROM
(select state, postal_code, email, display_size from PostalCode
join Household
ON postal_code = Household.FK_HouseHold_postal_code_PostalCode_postal_code
join TV
ON Household.email = TV.FK_tv_email_HouseHold_email) ALL_TV_IN_STATE
GROUP BY state
ORDER BY state ASC),
x2 AS (
SELECT state, display_type, maximum_resolution from PostalCode
JOIN Household
ON postal_code = HouseHold.FK_HouseHold_postal_code_PostalCode_postal_code
JOIN TV
ON HouseHold.email = TV.FK_tv_email_HouseHold_email)
select x1.state, display_type AS screen_type, maximum_resolution, FORMAT(Average_size,
'2.#') AS Average_size from x1
JOIN
x2
WHERE x1.state = '$State'
```

8. Extra Fridge / Freezer Report

When Users click to review 'Extra Fridge/ Freezer' report:

8.a

At the top of this report, a count of all households with more than one fridge or freezer should be displayed

- From the Freezer table, aggregate on email, count the occurances of households using their email.
- With the result above, count the number of emails

```
SELECT COUNT(email) AS num_household_multi_freezer FROM (
SELECT FK_Freezer_email_HouseHold_email AS email,
```

```
COUNT(FK_Freezer_email_HouseHold_email) AS num_freezer
FROM Freezer
GROUP by email) Multi_Freezer_Household
WHERE num_freezer > 1;
```

8.b

Underneath the count, a tabular listing will show the top ten states with a column for the state and a column for the count of households with multiple fridge/freezers in that state, sorted by household count descending. Additional columns after the household count will show the following values: the percentage of households with multiple fridge/freezers in that state with chest freezers, the percentage of households with multiple fridge/freezers in that state with an upright freezer, and the percentage of households with multiple fridge/freezers in that state with something else

- Collect the households that have multiple fridges/freezers
- Collect the household that have multiple fridges/freezers of type chest. Count the number of households
- Collect the household that have multiple fridges/freezers of type upright. Count the number of households
- Collect the household that have multiple fridges/freezers of other types. Count the number of households
- Left join the 4 tables, and compute the percentage for each respected type

```
select  Multi_Freezer_Household.state,
      Multi_Freezer_Household.num_household_multi_freezer,

100*Chest_Count.chest_houshold_count/Multi_Freezer_Household.num_household_multi_fr
eezer as chest_household_percentage,

100*Upright_Count.upright_householdd_count/Multi_Freezer_Household.num_household_m
ulti_freezer as upright_household_percentage,

100*Other_Count_Aggregated.other_household_count/Multi_Freezer_Household.num_house
hold_multi_freezer as other_household_percentage
FROM
(
-- Get the count of multi-fridge household
-- 4) aggregate the count of apperances of each state
select state, count(state) as num_household_multi_freezer from

-- 3) get the email and the state of those household
( select state, email
from PostalCode inner join

-- 2) get the email and the postal code of those household
(select Multi_Freezer_Household.email,
FK_HouseHold_postal_code_PostalCode_postal_code as postal_code
from HouseHold inner join
```

```
-- 1) aggreate to get number of freezer for each existing household
(select FK_Freezer_email_HouseHold_email as email,
count(FK_Freezer_email_HouseHold_email) as num_freezer
from Freezer
group by email) Multi_Freezer_Household
-- end of 1)

on HouseHold.email = Multi_Freezer_Household.email
where Multi_Freezer_Household.num_freezer  > 1) Multi_Freezer_Household_PostalCode
-- end of 2)

on PostalCode.postal_code = Multi_Freezer_Household_PostalCode.postal_code)
Multi_Freezer_Household_state
-- end of 3)

group by state
order by num_household_multi_freezer desc
limit  10 ) Multi_Freezer_Household

LEFT JOIN

(
select state, count(email) as chest_houshold_count FROM
(
-- 5) get the model type, state, and their count
select email, model_type, state, count(email) as model_type_individual_household_count
from (

-- 4) get the household, modeltype, and state
select FK_Freezer_email_HouseHold_email as email, model_type, state from Freezer inner
join

-- 3) get the email and the state of those household with multi freezers
( select state, email
from PostalCode inner join

-- 2) get the email and the postal code of those household with multi freezers
(select Multi_Freezer_Household.email,
FK_HouseHold_postal_code_PostalCode_postal_code as postal_code
from HouseHold inner join

-- 1) aggreate to get number of freezer for each existing household
(select FK_Freezer_email_HouseHold_email as email,
count(FK_Freezer_email_HouseHold_email) as num_freezer
from Freezer
group by email) Multi_Freezer_Household
-- end of 1)

on HouseHold.email = Multi_Freezer_Household.email
```

```
where Multi_Freezer_Household.num_freezer  > 1) Multi_Freezer_Household_PostalCode
-- end of 2)

on PostalCode.postal_code = Multi_Freezer_Household_PostalCode.postal_code)
Multi_Freezer_Household_State
--  end of 3)

on Freezer.FK_Freezer_email_HouseHold_email = Multi_Freezer_Household_State.email )
Multi_Freezer_Household_Type_State
-- end of 4)
group by email, model_type, state) Model_Type_Count
-- end of 5
group by model_type, state
having model_type = 'chest'
) Chest_Count

ON Multi_Freezer_Household.state = Chest_Count.state

LEFT JOIN
(
select state, count(email) as upright_householdd_count FROM
(
-- 5) get the model type, state, and their count
-- select model_type, state, count(email) as model_type_household_count from (
select email, model_type, state, count(email) as model_type_individual_household_count
from (

-- 4) get the household, modeltype, and state
select FK_Freezer_email_HouseHold_email as email, model_type, state from Freezer inner
join

-- 3) get the email and the state of those household with multi freezers
( select state, email
from PostalCode inner join

-- 2) get the email and the postal code of those household with multi freezers
(select Multi_Freezer_Household.email,
FK_HouseHold_postal_code_PostalCode_postal_code as postal_code
from HouseHold inner join

-- 1) aggreate to get number of freezer for each existing household
(select FK_Freezer_email_HouseHold_email as email,
count(FK_Freezer_email_HouseHold_email) as num_freezer
from Freezer
group by email) Multi_Freezer_Household
-- end of 1)

on HouseHold.email = Multi_Freezer_Household.email
where Multi_Freezer_Household.num_freezer  > 1) Multi_Freezer_Household_PostalCode
-- end of 2)
```

```
on PostalCode.postal_code = Multi_Freezer_Household_PostalCode.postal_code)
Multi_Freezer_Household_State
--  end of 3)

on Freezer.FK_Freezer_email_HouseHold_email = Multi_Freezer_Household_State.email )
Multi_Freezer_Household_Type_State
-- end of 4)
group by email, model_type, state) Model_Type_Count

group by model_type, state
having model_type = 'upright'
) Upright_Count

ON Multi_Freezer_Household.state = Upright_Count.state
LEFT JOIN
(
select state, count( DISTINCT email) as other_household_count FROM
(
select email, model_type, state, count(email) as model_type_individual_household_count
from (
-- 4) get the household, modeltype, and state
select FK_Freezer_email_HouseHold_email as email, model_type, state from Freezer inner
join

-- 3) get the email and the state of those household with multi freezers
( select state, email
from PostalCode inner join

-- 2) get the email and the postal code of those household with multi freezers
(select Multi_Freezer_Household.email,
FK_HouseHold_postal_code_PostalCode_postal_code as postal_code
from HouseHold inner join

-- 1) aggreate to get number of freezer for each existing household
(select FK_Freezer_email_HouseHold_email as email,
count(FK_Freezer_email_HouseHold_email) as num_freezer
from Freezer
group by email) Multi_Freezer_Household
-- end of 1)

on HouseHold.email = Multi_Freezer_Household.email
where Multi_Freezer_Household.num_freezer  > 1) Multi_Freezer_Household_PostalCode
-- end of 2)

on PostalCode.postal_code = Multi_Freezer_Household_PostalCode.postal_code)
Multi_Freezer_Household_State
-- end of 3)
on Freezer.FK_Freezer_email_HouseHold_email = Multi_Freezer_Household_State.email )
Multi_Freezer_Household_Type_State
```

```
-- end of 4)
group by email, model_type, state
having model_type not in ('chest', 'upright')
) household_other_freezer
group by state
) Other_Count_Aggregated
ON Multi_Freezer_Household.state = Other_Count_Aggregated.state
```

9. **Laundry Center Report**
    - SQL Query 1 for maximum washer loading_type count and maximum dryer heat_source count per state:

This query will first obtain the washer loading type and its count (frequency) grouped with the state. Then obtain the highest frequency with its related loading type that can be found in each state. Then we repeat the same process with the dryer heat source to obtain the highest frequency with its related heat source that can be found in each state. Left join with the PostalCode table with the loading type that has the highest frequency on the state and then left join with the dryer heat source with the highest frequency to formulate the result.

```
WITH WasherFrequencyPerState AS (SELECT w.loading_type, COUNT(w.loading_type) as
frequency, p.state FROM Washer w
   JOIN HouseHold h ON w.FK_Washer_email_HouseHold_email = h.email
   JOIN PostalCode p ON h.FK_HouseHold_postal_code_PostalCode_postal_code =
p.postal_code
   GROUP BY w.loading_type, p.state ORDER BY p.state, frequency DESC)
,
WasherHighestFrequencyPerState AS (SELECT MAX(w.frequency) AS highest_frequency,
w.state FROM WasherFrequencyPerState w GROUP BY w.state),
WasherTopPerState AS (SELECT wfps.loading_type, wfps.frequency, wfps.state FROM
WasherFrequencyPerState wfps JOIN WasherHighestFrequencyPerState whfps ON
whfps.state = wfps.state WHERE whfps.highest_frequency = wfps.frequency),
DryerFrequencyPerState AS (SELECT d.heat_source, COUNT(d.heat_source) as frequency,
p.state FROM Dryer d
   JOIN HouseHold h ON d.FK_Dryer_email_HouseHold_email = h.email
   JOIN PostalCode p ON h.FK_HouseHold_postal_code_PostalCode_postal_code =
p.postal_code
   GROUP BY d.heat_source, p.state ORDER BY p.state, frequency DESC)
,
DryerHighestFrequencyPerState AS (SELECT MAX(d.frequency) AS highest_frequency,
d.state FROM DryerFrequencyPerState d GROUP BY d.state),
DryerTopPerState AS (SELECT dfps.heat_source, dfps.frequency, dfps.state FROM
DryerFrequencyPerState dfps JOIN DryerHighestFrequencyPerState dhfps ON dhfps.state =
dfps.state WHERE dhfps.highest_frequency = dfps.frequency)
SELECT DISTINCT(p.state), wtps.loading_type, dtps.heat_source FROM PostalCode p
LEFT JOIN WasherTopPerState wtps
ON p.state = wtps.state
LEFT JOIN DryerTopPerState dtps
ON p.state = dtps.state
ORDER BY p.state;
```

- SQL Query 2 for count of household with only washer but not dryer:

This query will first obtain the washer count for each household then obtain its dryer count then join them together on the household. The count in each state, how many households have washer but with no dryer.

```
WITH WasherPerHouseHold AS (SELECT COUNT(h.email) as
washer_count_per_household, h.email, p.state FROM HouseHold h
JOIN PostalCode p ON h.FK_HouseHold_postal_code_PostalCode_postal_code =
p.postal_code
RIGHT JOIN Washer w ON w.FK_Washer_email_HouseHold_email = h.email
GROUP BY h.email),
DryerPerHouseHold AS (SELECT COUNT(h.email) as dryer_count_per_household, h.email,
p.state FROM HouseHold h
JOIN PostalCode p ON h.FK_HouseHold_postal_code_PostalCode_postal_code =
p.postal_code
RIGHT JOIN Dryer d ON d.FK_Dryer_email_HouseHold_email = h.email
GROUP BY h.email),
WasherDryerPerHouseHold AS (SELECT h.email, wph.washer_count_per_household,
dph.dryer_count_per_household, p.state FROM HouseHold h
JOIN PostalCode p ON p.postal_code =
h.FK_HouseHold_postal_code_PostalCode_postal_code
LEFT JOIN WasherPerHouseHold wph ON h.email = wph.email
LEFT JOIN DryerPerHouseHold dph ON h.email = dph.email)
SELECT wdph.state, COUNT(wdph.email) count_household_with_washer_no_dryer
FROM WasherDryerPerHouseHold wdph
WHERE wdph.dryer_count_per_household IS NULL AND
wdph.washer_count_per_household IS NOT NULL
GROUP BY wdph.state
ORDER BY count_household_with_washer_no_dryer AND wdph.state;
```

10. Bathroom Statistics Report

- Query 1: The minimum (as an integer), average (as a decimal number rounded up to the tenths decimal point), and maximum (as an integer) count of all bathrooms per household.
    - Query for overlapping union of e-mails in Full and Half tables.
    - Group based on e-mail
    - Get table of e-mails and bathroom counts from above
    - Get table with minimal average and max count of bathrooms from above
- Query 2: The minimum (as an integer), average (as a decimal number rounded up to the tenths decimal point), and maximum (as an integer) count of half bathrooms per household.
    - Query for e-mails in Half table.
    - Group based on e-mail
    - Get table of e-mails and bathroom counts from above
    - Get table with minimal average and max count of bathrooms from above
- Query 3: The minimum (as an integer), average (as a decimal number rounded up to the tenths decimal point), and maximum (as an integer) count of full bathrooms per household.
    - Query for e-mails in Full table.

    - Group based on e-mail
    - Get table of e-mails and bathroom counts from above
    - Get table with minimal average and max counts of bathrooms from above
- Query 4: The minimum (as an integer), average (as a decimal number rounded up to the tenths decimal point), and maximum (as an integer) count of commodes per household.
    - Query for overlapping union of e-mails, commode numbers in Half and Full tables.
    - Group based on e-mail
    - Get table of e-mails and commode sums
    - Get table with minimal average and max count of commodes from above
- Query 5: The minimum (as an integer), average (as a decimal number rounded up to the tenths decimal point), and maximum (as an integer) count of sinks per household.
    - Query for overlapping union of e-mails, sinks numbers in Half and Full tables.
    - Group based on e-mail
    - Get table of e-mails and sink sums
    - Get table with minimal average and max count of sinks from above
- Query 6: The minimum (as an integer), average (as a decimal number rounded up to the tenths decimal point), and maximum (as an integer) count of bidets per household.
    - Query for overlapping union of e-mails, bidet numbers in Half and Full tables.
    - Group based on e-mail
    - Get table of e-mails and bidet sums
    - Get table with minimal average and max count of bidet from above
- Query 7: The minimum (as an integer), average (as a decimal number rounded up to the tenths decimal point), and maximum (as an integer) count of bathtubs per household.
    - Query for e-mails, bathtubs in Full table.
    - Group based on e-mail
    - Get table of e-mails and bathtub counts from above
    - Get table with minimal average and max count of bathtubs from above
- Query 8: The minimum (as an integer), average (as a decimal number rounded up to the tenths decimal point), and maximum (as an integer) count of showers per household.
    - Query for e-mails, showers in Full table.
    - Group based on e-mail
    - Get table of e-mails and shower counts from above
    - Get table with minimal average and max count of showers from above
- Query 9: The minimum (as an integer), average (as a decimal number rounded up to the tenths decimal point), and maximum (as an integer) count of tub/showers per household.
    - Query for e-mails, tub/showers in Full table.
    - Group based on e-mail
    - Get table of e-mails and tub/shower counts from above
    - Get table with minimal average and max count of tub/showers from above
- Query 10: Which state has the most bidets (count of all bidets as an integer), and how many.
    - Get e-mails with bidet counts from overlapping union of Full and Half.
    - Join with tables Household and Postalcode based on appropriate key(emails, postal codes)
    - Group based on state.

    - Get table of states with bidet counts
    - Sort in descending order.
    - Display first entry only
- Query 11: Which postal code has the most bidets (count of all bidets as an integer), and how many.
    - Get e-mails with bidet counts from overlapping union of Full and Half.
    - Join with tables Household and Postalcode based on appropriate key(emails, postal codes),
    - Group based on zipcode.
    - Get zipcodes of states with bidet counts
    - Sort in descending order.
    - Display first entry only
- Query 12: How many households (count as an integer), have only a single, primary bathroom, and no other bathrooms.
    - Query for overlapping union of Full and Half with e-mails columns.
    - From result select rows based on condition that primary bathroom with given e-mail exists in Full
    - Group based on e-mail
    - Get table of e-mails with bathroom counts from above.
    - Count number of e-mails with bath count equal to 1 in above.

```
-- The minimum (as an integer), average (as a decimal number rounded up to the tenths decimal point), and maximum (as an integer) count of all bathrooms per household

SELECT MIN(Cnt) AS min_bath_perhousehold, AVG(Cnt) AS avg_bath_perhousehold, MAX(Cnt) AS max_bath_perhousehold FROM (SELECT FK_Half_email_HouseHold_email, Count(*) as 'Cnt' FROM (SELECT number, FK_Half_email_HouseHold_email FROM Half UNION ALL
SELECT number, FK_Full_email_HouseHold_email FROM Full) Bathrooms
GROUP BY FK_Half_email_HouseHold_email) Bathrooms_cnt
-- The minimum (as an integer), average (as a decimal number rounded up to the tenths decimal point), and maximum (as an integer) count of half bathrooms per household
SELECT MIN(Cnt) AS min_halfbath_perhousehold, AVG(Cnt) AS avg_halfbath_perhousehold, MAX(Cnt) AS max_halfbath_perhousehold FROM (SELECT FK_Half_email_HouseHold_email, Count(*) as 'Cnt' FROM (SELECT number, FK_Half_email_HouseHold_email FROM Half ) Bathrooms
GROUP BY FK_Half_email_HouseHold_email) Bathrooms_cnt
-- The minimum (as an integer), average (as a decimal number rounded up to the tenths decimal point), and maximum (as an integer) count of full bathrooms per household
SELECT MIN(Cnt) AS min_fullbath_perhousehold, AVG(Cnt) AS avg_fullbath_perhousehold, MAX(Cnt) AS max_fullbath_perhousehold FROM
```

```
(SELECT FK_Full_email_HouseHold_email, Count(*) as 'Cnt' FROM (SELECT
number, FK_Full_email_HouseHold_email FROM Full ) Bathrooms
GROUP BY FK_Full_email_HouseHold_email) Bathrooms_cnt
-- The minimum (as an integer), average (as a decimal number rounded up to the
tenths decimal point), and maximum (as an integer) count of commodes per
household
SELECT MIN(commode_count) AS min_commode_perhousehold,
AVG(commode_count) AS avg_commode_perhousehold, MAX(commode_count) AS
max_commode_perhousehold FROM (SELECT FK_Half_email_HouseHold_email,
Sum(commode) as 'commode_count' FROM (SELECT number,
FK_Half_email_HouseHold_email, commode FROM Half UNION ALL
SELECT number, FK_Full_email_HouseHold_email, commode FROM Full)
Bathrooms
GROUP BY FK_Half_email_HouseHold_email) commodes_cnt
-- The minimum (as an integer), average (as a decimal number rounded up to the
tenths decimal point), and maximum (as an integer) count of sinks per household
SELECT MIN(sink_count) AS min_sink_perhousehold, AVG(sink_count) AS
avg_sink_perhousehold, MAX(sink_count) AS max_sink_perhousehold FROM
(SELECT FK_Half_email_HouseHold_email, Sum(sink) as 'sink_count' FROM
(SELECT number, FK_Half_email_HouseHold_email, sink FROM Half UNION ALL
SELECT number, FK_Full_email_HouseHold_email, sink FROM Full) Bathrooms
GROUP BY FK_Half_email_HouseHold_email) sink_cnt
-- The minimum (as an integer), average (as a decimal number rounded up to the
tenths decimal point), and maximum (as an integer) count of bidets per household
SELECT MIN(bidet_count) AS min_bidet_perhousehold, AVG(bidet_count) AS
avg_bidet_perhousehold, MAX(bidet_count) AS max_bidet_perhousehold FROM
(SELECT FK_Half_email_HouseHold_email, Sum(bidet) as 'bidet_count' FROM
(SELECT number, FK_Half_email_HouseHold_email, bidet FROM Half UNION ALL
SELECT number, FK_Full_email_HouseHold_email, bidet FROM Full) Bathrooms
GROUP BY FK_Half_email_HouseHold_email) bidet_cnt
-- The minimum (as an integer), average (as a decimal number rounded up to the
tenths decimal point), and maximum (as an integer) count of bathtubs per household
SELECT MIN(bathtub_count) AS min_bathtub_perhousehold, AVG(bathtub_count)
AS avg_bathtub_perhousehold, MAX(bathtub_count) AS max_bathtub_perhousehold
FROM (SELECT FK_Full_email_HouseHold_email, Sum(bathtub) as 'bathtub_count'
FROM (
SELECT number, FK_Full_email_HouseHold_email, bathtub FROM Full) Bathrooms
GROUP BY FK_Full_email_HouseHold_email) bathtub_cnt
-- The minimum (as an integer), average (as a decimal number rounded up to the
tenths decimal point), and maximum (as an integer) count of showers per household
SELECT MIN(shower_count) AS min_shower_perhousehold, AVG(shower_count) AS
avg_shower_perhousehold, MAX(shower_count) AS max_shower_perhousehold
FROM (SELECT FK_Full_email_HouseHold_email, Sum(shower) as 'shower_count'
FROM (
SELECT number, FK_Full_email_HouseHold_email, shower FROM Full) Bathrooms
GROUP BY FK_Full_email_HouseHold_email) shower_cnt
-- The minimum (as an integer), average (as a decimal number rounded up to the
tenths decimal point), and maximum (as an integer) count of tub/showers per
household
```

```
SELECT MIN(tub_shower_count) AS min_tub_shower_perhousehold,
AVG(tub_shower_count) AS avg_tub_shower_perhousehold,
MAX(tub_shower_count) AS max_tub_shower_perhousehold FROM (SELECT
FK_Full_email_HouseHold_email, Sum(tub_shower) as 'tub_shower_count' FROM (
SELECT number, FK_Full_email_HouseHold_email, tub_shower FROM Full)
Bathrooms
GROUP BY FK_Full_email_HouseHold_email) tub_shower_cnt
-- Which state has the most bidets (count of all bidets as an integer), and how many
(SELECT state as 'state_with_most_bidets', SUM(bidet) as 'bidet_count' FROM
((SELECT PostalCode.state AS 'state' , HouseHold.email, Full.bidet FROM
PostalCode
JOIN HouseHold ON
HouseHold.FK_HouseHold_postal_code_PostalCode_postal_code=PostalCode.posta
l_code
JOIN Full ON HouseHold.email=Full.FK_Full_email_HouseHold_email) UNION ALL
(SELECT PostalCode.state, HouseHold.email, Half.bidet FROM PostalCode
JOIN HouseHold ON
HouseHold.FK_HouseHold_postal_code_PostalCode_postal_code=PostalCode.posta
l_code
JOIN Half ON HouseHold.email=Half.FK_Half_email_HouseHold_email)) state_bidets
GROUP BY state) ORDER BY `bidet_count` desc  LIMIT 1;

-- Which postal code has the most bidets (count of all bidets as an integer), and how
many
(SELECT postal_code as 'postal_with_most_bidets', SUM(bidet) as 'bidet_count'
FROM ((SELECT PostalCode.postal_code AS 'postal_code' , HouseHold.email,
Full.bidet FROM PostalCode
JOIN HouseHold ON
HouseHold.FK_HouseHold_postal_code_PostalCode_postal_code=PostalCode.posta
l_code
JOIN Full ON HouseHold.email=Full.FK_Full_email_HouseHold_email) UNION ALL
(SELECT PostalCode.postal_code, HouseHold.email, Half.bidet FROM PostalCode
JOIN HouseHold ON
HouseHold.FK_HouseHold_postal_code_PostalCode_postal_code=PostalCode.posta
l_code
JOIN Half ON HouseHold.email=Half.FK_Half_email_HouseHold_email)) state_bidets
GROUP BY postal_code) ORDER BY `bidet_count` desc  LIMIT 1;
-- How many households (count as an integer), have only a single, primary bathroom,
and no other bathrooms
(SELECT COUNT(*) as 'single_bath_household_cnt' FROM (SELECT
FK_Full_email_HouseHold_email, Count(*) as 'Cnt' FROM (SELECT number,
FK_Full_email_HouseHold_email FROM Full UNION ALL
SELECT number, FK_Half_email_HouseHold_email FROM Half) Bathrooms WHERE
EXISTS (SELECT * FROM Full WHERE

(Full.FK_Full_email_HouseHold_email=Bathrooms.FK_Full_email_HouseHold_email
AND Full.is_primary=1))
GROUP BY FK_Full_email_HouseHold_email HAVING Cnt=1) single_bath_counts)
```

11. Household Average by Radius Report

Note: Big picture approach is two steps. The first step is to update *householdInfoDict*, heatSourceDict[zip][heat_source], second step is to display information.

- Do nothing until the user hits the submit button.
- Parse input string and test against list of zipcodes provided (go to step1).

Step 1: User Input PostalCode validation

```
SELECT * FROM postalCode
WHERE exists
(SELECT postal_code from PostalCode where postal_code = '$postal_code')
```

If results back from the above query is null, display error message to the user. Give the user option to re-enter.

- If get valid results from step1, go to step 2.

Step 2: Retrieve Lat and Lon

```
SELECT latitude, longitude FROM PostalCode
WHERE postal_code = '$UserInputPostalCode'
```

From valid user input postalCode, query latitude and longitude for that PostalCode.

Store results in variables '$Lon' and '$Lat ' for use in step 3.

- Collect variables from step 2 result and user input for Distance, go to step 3.

Step 3. Radius Report Query

Use Lat and Lon from step 2, together with user input Distance/Radius ('$Distance'), for this query.

(abstract code for this query:

- List all postal code within the distance from the input coordinate
- List all households for the postalCode from previous step
- Get the occupant, bedroom, bathroom information for the households
- Calculate the radio of commodes to occupant per household and store it in a column
- Calculate number of bathrooms per household and store it in a column
- Count total number of appliances a household owns and store in a column
- Collect/count heat sources for each
- Combine the previous three intermediate results in one table with household, postalCode and occupants' information
- Calculate average bathroom, bedroom, occupant, appliances count, per household, for the number of households belonging to a PostalCode
- Aggregate the results in a table group by PostalCode)

```
SELECT postal_code, AVG(DistanceFromInputLocation), AVG(occupant), AVG(bedroom),
AVG(NumberOfBathroom), AVG(RatioOfCommodeToOccupant)
FROM
(With x0 AS (select FK_Freezer_email_HouseHold_email AS Email,
(NumberOfApp.A+NumberOfApp.B+NumberOfApp.C+NumberOfApp.D+NumberOfApp.E) AS
Total From
(With FreezerOwnedPerHousehold As (select FK_Freezer_email_HouseHold_email, count(*)
AS A from FREEZER
group by FK_Freezer_email_HouseHold_email),
CookerOwnedPerHousehold AS (select FK_Cooker_email_HouseHold_email, count(*) AS B
from COOKER
group by FK_Cooker_email_HouseHold_email),
WasherOwnedPerHousehold AS (select FK_Washer_email_HouseHold_email, count(*) AS C
from WASHER
group by FK_Washer_email_HouseHold_email),
DryerOwnedPerHousehold AS (select FK_Dryer_email_HouseHold_email, count(*) AS D
from DRYER
group by FK_Dryer_email_HouseHold_email),
TVOwnedPerHousehold AS (select FK_TV_email_HouseHold_email, count(*) AS E from TV
group by FK_TV_email_HouseHold_email)
select * from FreezerOwnedPerHousehold
left join
CookerOwnedPerHousehold
ON FreezerOwnedPerHousehold.FK_Freezer_email_HouseHold_email =
CookerOwnedPerHousehold.FK_Cooker_email_HouseHold_email
left join
WasherOwnedPerHousehold
ON FreezerOwnedPerHousehold.FK_Freezer_email_HouseHold_email =
WasherOwnedPerHousehold.FK_Washer_email_HouseHold_email
left join
DryerOwnedPerHousehold
ON FreezerOwnedPerHousehold.FK_Freezer_email_HouseHold_email =
DryerOwnedPerHousehold.FK_Dryer_email_HouseHold_email
left join
TVOwnedPerHousehold
ON FreezerOwnedPerHousehold.FK_Freezer_email_HouseHold_email =
TVOwnedPerHousehold.FK_TV_email_HouseHold_email) AS NumberOfApp),
x1 AS
(Select postal_code, email, occupant, bedroom, D, Full.number AS fullNumber, Half.number
AS halfNumber, has_gas_heat_source, has_electric_heat_source,
has_microwave_heat_source, Cooktop.heat_source, FULL.commode AS FullCommode,
HALF.commode As HalfCommode from
(Select postal_code, city, latitude, longitude,
   acos(sin('$Lat') * sin(latitude) + cos('$Lat') * cos(latitude) * cos(longitude - ('$Lon'))) *
3958.8 As D
From PostalCode
```

```
Where acos(sin('$Lat') * sin(latitude) + cos('$Lat') * cos(latitude) * cos(longitude - ('$Lon'))) *
3958.8 <= '$Distance') AS PostalCode_Within_Distance
left Join Household
ON PostalCode_Within_Distance.postal_code =
Household.FK_HouseHold_postal_code_PostalCode_postal_code
left Join Oven
ON Household.email = Oven.FK_oven_email_HouseHold_email
left Join Cooktop
   On Household.email = Cooktop.FK_cooktop_email_Household_email
   left Join FULL
ON Household.email = FULL.FK_Full_email_HouseHold_email
left Join HALF
ON Household.email = HALF.FK_Half_email_Household_email),
x2 AS (select email, FORMAT(x1.occupant/coalesce(x1.FullCommode+x1.HalfCommode,
x1.FullCommode, x1.HalfCommode, 0), '2:#') AS RatioOfCommodeToOccupant from x1),
x3 AS (select email, (x1.fullNumber+x1.halfNumber) AS NumberOfBathroom from x1)
select  x1.postal_code, X1.D As DistanceFromInputLocation, occupant, bedroom,
x3.NumberOfBathroom, RatioOfCommodeToOccupant, x0.Total AS NumberOfAppliance
from x1
join x2
on x1.email = x2.email
join x3
on x1.email = x3.email
join x0
on x1.email = x0.email) AS STATISTICS
GROUP BY postal_code
```

- Present above information in 5 columns: zip, distance, average occupant, bedroom, bathroom, commodes-per-occupant, average appliance, most common heat_source.