Table of Contents

Abstract Code

Enter Information

Email Address

Abstract Code

1. User clicks ***Enter my household info*** button on the **Main Menu**.
2. Display **Email Address** form.
3. While the *email* input field is not inputted or valid:
    a. *email* ('$email') input field is filled and validated
    b. If the ***Submit*** button is clicked:
        i. Run the **Verify Email** task:
            1. If '$email' is already found in the Household entity

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

                a. Display an error message indicating that the email is already in use
                b. Erase the input and return to step 2.
            2. Else
                a. Store '$email' as a session variable.

Postal Code

Abstract Code

1. Display **Postal Code Form**.
2. While *postal code* input field is not inputted or valid:
    a. *postal code* ('$postal_code') field is inputted
    b. If ***Submit*** button is clicked:
        i. Run the **Verify Postal Code** task:
            1. If the '$postal_code' is not found in the Location entity

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

                a. Display an error message indicating that the *postal code* input was invalid.
                b. Erase the input and return to step 1.
            2. Else
                a. Display confirmation page with the Location.postal_code, Location.city and Location.state for the matching record found in the Location entity with ***Yes*** button and ***No*** button.

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

                b. If ***No*** button is clicked:
                    i. Jump to step 1.
                c. Else, if ***Yes*** button is clicked:
                    i. Store '$postal_code' as a session variable.

Phone Number

Abstract Code

1. Display **Phone Number Form**.
2. Display prompt with ***Yes*** or ***No*** toggle to enter a phone number.
    a. If ***Yes*** toggle is clicked:
        i. While the *area code* input field is not inputted, and the *phone number* is not inputted, and the *phone type* is not inputted
            1. The *area code* ('$area_code') input field is filled by user.
            2. The *number* ('$phone_number') input field is filled by user.
            3. The *phone type* ('$phone_type') is selected from drop down menu.
            4. If the *area code*, *phone number*, and *phone type* inputs are all filled:
                a. Jump to step 2.a.ii.
            5. Else
                a. Jump to step 2a
        ii. If the ***Next*** button is clicked:
            1. Run the **Verify Phone Number** task:
                a. If the combination of the '$area_code' and '$phone_number' input fields is found in the Phone_Number entity:

```
SELECT
CONCAT(area_code, phone_number) as phone
FROM Phone_Number
WHERE area_code = '$area_code'
AND phone_number = '$phone_number';
```

                    i. Display an error message stating that '$phone_number' is already in use.
                    ii. Erase the input and return to step 1.
                b. Else:
                    i. Store '$area_code' and '$phone_number' as session variables

Household Information

Abstract Code

1. Display **Household Details Form**.
    a. User selects *home type* ('$home_type') from dropdown.
    b. *square footage* ('$square_footage') input field is filled by the user
    c. *occupant count* ('$occupant_count') input field is filled by the user
    d. *bedroom count* ('$bedroom_count') input field is filled by the user
    e. If the ***Next*** button is clicked:
        i. If *square footage*, *occupant count*, and *bedroom count* input fields are not inputted or are invalid:
            1. Display an error message indicating that *square footage*, *occupant count*, and *bedroom count* input fields are invalid.
            2. Erase inputs and return to step 1.
        ii. Else
            1. Store '$home_type', '$square_footage', '$occupant_count', and '$bedroom_count' as session variables.
            2. Jump to step 2.
2. If the ***Next*** button is clicked on the **Household Details Form**:
    a. Add the household information to the Household and Phone_Number entities by running the **Add Household Details** task:
        i. Run **Add Household Information** task: Add a new household in the Household entity with the '$email' as the Household.email, the '$home_type' as the Household.home_type, '$square_footage' as the Household.square_footage, '$occupant_count' as the Household.occupant_count, and '$bedroom_count' as the Household.bedroom_count, associate the household with the record in the Location entity where Location.postal_code == '$postal_code'

```
INSERT INTO Household
(
      email,
      home_type,
      square_footage,
      occupant_count,
      bedroom_count,
      postal_code
)
VALUES
(
      '$email',
      '$home_type',
      '$square_footage',
      '$occupant_count',
      '$bedroom_count',
      '$postal_code'
);
```

ii. If use opted in to enter '$phone_number'
   1. Run **Add Phone Number** task: Add the '$area_code' and '$phone_number', '$phone_type', and '$email' as the phone number to the Phone_Number entity for the household with the matching email from the Household entity. Remove the hyphen(s) if they appear in '$phone_number'.

```
INSERT INTO Phone_Number
(
      area_code,
      phone_number,
      phone_type,
      email
)
VALUES
(
      '$area_code',
      REPLACE('$phone_number','-',''),
      '$phone_type',
      '$email'
);
```

Add Bathroom

Abstract Code

1. While the Hemkraft site is still open:
    a. Start with '$bathroom_count' = 0
    b. Display **Add Bathroom Form** with options of 'Half' or 'Full' for the *bathroom type*
    c. While ***Add*** button is not clicked, do nothing.
    d. If user selects 'Half' as *bathroom type* ('$bathroom_type') in **Add Bathroom Form**.
        i. *sink count* ('$sink_count') input field is filled
        ii. *commode count* ('$commode_count') input field is filled
        iii. *bidet count* ('$bidet_count') input field is filled
        iv. *name* ('$name') input field is filled
        v. If ***Add*** button is clicked, do the following:
            1. If any of the *sink count*, *commode count*, *bidet count* fixtures are negative
                a. Display **Add Bathroom Form** with error message indicating that none of the *sink count*, *commode count*, *bidet count* input fields can be negative.
                b. Erase input and jump to step 1c.
            2. Else if the sum of count of the '$sink_count', '$commode_count', and '$bidet_count' is not greater than 0
                a. Display **Add Bathroom Form** with error message indicating that the total of the sum of *sink count*, *commode count*, *bidet count* should be greater than zero.
                b. Erase input and jump to step 1c.
            3. Else
                a. Increment '$bathroom_count' by 1
                b. Run **Add Half or Full Bathroom** task: Add a bathroom in the Half entity for the household where Household.email == '$email' with the '$bidet_count' as Half.bidet_count, '$sink_count' as Half.sink_count, '$commode_count' as Half.commode_count, and '$name' as Half.name if the user provided the optional half bathroom name, '$bathroom_count' as Half.bathroom_id to Half entity.
                c. Jump to step 1f.

```
INSERT INTO Half (
    bathroom_id,
    email,
    sink_count,
    bidet_count,
    commode_count,
    `name`
  )
VALUES (
    '$bathroom_count',
    '$email',
    '$sink_count',
    '$bidet_count',
    '$commode_count',
    '$name'
  );
```

e. Else if user selects 'Full' as *bathroom type* ('$bathroom_type') in **Add Bathroom Form**.
   i. For all bathrooms in the Bathroom entity where the bathroom belongs to the household where Household.email == '$email' and the *bathroom type* is `Full`
      1. If `Full`.is_primary == 1,

```
SELECT *
FROM `Full`
WHERE is_primary = 1
AND email = '$email';
```

         a. Disable the ***This bathroom is a primary bathroom*** checkbox
   ii. While ***Add*** button is not clicked, do nothing.
   iii. User enters *sink count* ('$sink_count'), *commode count* ('$commode_count'), *bidet count* ('$bidet_count'), *bathtub count* ('$bathtub_count'), *shower/tub count* ('$showertub_count'), *shower count* ('$shower_count') and if the checkbox is enabled, the user also enters if bathroom *is primary* ('$is_primary').
   iv. When ***Add*** button is clicked, do the following:
      1. If any of the of the '$sink_count', '$commode_count', '$bidet_count', '$bathtub_count', '$showertub_count', or '$shower_count' values are negative
         a. Display **Add Bathroom Form** with error message indicating that none of the *sink count*, *commode count*, *bidet count*, *bathtub*

count, *shower/tub count*, or *shower count* input fields can be negative.

b. Erase input and jump to step 1d

2. Else if the sum of the '$bathtub_count', '$showertub_count', and '$shower_count' is not greater than zero
   a. Display **Add Bathroom Form** with error message indicating that the sum of the *bathtub count*, *shower/tub count*, and *shower count* input fields must be greater than zero.
   b. Erase input and jump to step 1d
3. Else
   a. Increment '$bathroom_count' by 1
   b. Run **Add Half or Full Bathroom** task: Add a bathroom in the `Full` entity for the household where Household.email == '$email' with the '$bidet_count' as `Full`.bidet_count, '$sink_count' as `Full`.sink_count, '$commode_count' as `Full`.commode_count, '$bathtub_count' as `Full`.BathtubCount, '$tub_shower_count' as `Full`.tub_shower_count, '$bathtub_count' as `Full`.bathtub_count, `Full`.bathroom_id as '$bathroom_count' to `Full` entity .

```
INSERT INTO `Full` (
    bathroom_id,
    email,
    sink_count,
    bidet_count,
    commode_count,
    is_primary,
    bathtub_count,
    shower_count,
    tub_shower_count
  )
VALUES (
    '$bathroom_count’,
    '$email’,
    '$sink_count',
    '$bidet_count',
    '$commode_count',
    '$is_primary',
    '$bathtub_count',
    '$shower_count',
    '$tub_shower_count'
  );
```

c. Jump to step 1f.

f. Run **Bathroom Listing** task:

i. For each record in `Full` and Half (Bathroom) entities for the household where Household.email == '$email':

1. Display the Bathroom.bathroom_id
2. If the Bathroom.bathroom_id is in the `Full` entity
   a. Display "Full" for the bathroom type and "Yes" if `Full`.is_primary else display an empty string for if the bathroom is a primary bathroom
3. Else if the Bathroom.bathroom_id is in the Half entity
   a. Display "Half" for the bathroom type and an empty string for if the bathroom is a primary bathroom.

```
SELECT bathroom_id as "Bathroom #",
  "Full" as "Type",
  CASE
    WHEN is_primary then 'Yes'
    ELSE ''
  END as "Primary"
FROM `Full`
WHERE email = '$email'
UNION
SELECT bathroom_id as "Bathroom #",
  "Half" as "Type",
  "" as "Primary"
FROM Half
WHERE email = '$email'
ORDER BY 'Bathroom #';
```

ii. Jump to step 1g.

g. Display a button to ***Add another bathroom***, and a ***Next*** button.

i. If the ***Add another bathroom*** button is clicked,

1. Display **Add Bathroom Form**
2. Jump to step 1a.

ii. Else if the ***Next*** button is clicked,

1. Display **Add Appliance Form**.

2. If the user closes the Hemkraft website or the Hemkraft website unexpectedly crashes before a bathroom is added:

a. Clear all partial data for the household where the '$email' == Household.email from the Household and Phone_Number entities.

```
-- Clear partial data in Household

DELETE FROM Phone_Number
WHERE email='$email';

DELETE FROM Household
WHERE email='$email';
```

Add Appliance

Abstract Code

1. While the Hemkraft website is still open:
    a. Start with '$appliance_count' = 0
    b. Populate the *appliance type* dropdown with 'Refrigerator/Freezer', 'Cooker', 'Washer', 'Dryer', and 'TV'.
    c. Populate the *manufacturer name* input field;

```
SELECT manufacturer_name from Manufacturer;
```

    d. While the *appliance type* is not selected and *manufacturer name* is not selected:
        i. *appliance type* ('$appliance_type') is selected from dropdown by the user
        ii. *manufacturer name* ('$manufacturer_name') is selected from dropdown by the user
        iii. *model name* ('$model_name') input field is optionally filled by the user.
        iv. If the *appliance type* and *manufacturer name* fields are selected:
            1. Jump to step 1d.
        v. Else
            1. Jump to step 1c.
    e. If '$appliance_type' == 'Cooker':
        i. Display **Add Appliance – Cooker Form**.
        ii. While ***Add*** button is not clicked, do nothing.
        iii. If ***oven*** is checked,
            1. The user selects one or more *oven heat source* ('$oven_heat_source') options
            2. The user selects the *oven type* ('$oven_type') from the dropdown options of 'Convection' or 'Conventional'
        iv. If ***cooktop*** is checked,
            1. The user selects the *cooktop heat source* ('$cooktop_heat_source') from the dropdown.
        v. If ***Add*** button is clicked, do the following:
            1. If ***oven*** is checked and *oven heat source* is selected,
                a. Increment '$appliance_count' by 1
                b. Run the **Add Appliance Subtype** task:

```
-- Create the Cooker
INSERT INTO Cooker
(
      appliance_id,
      email,
      model_name,
      manufacturer_name
)
VALUES
(
      '$appliance_count',
      '$email',
      '$model_name',
      '$manufacturer_name'
);


-- Create the Oven with the same appliance_id
INSERT INTO Oven
(
      appliance_id,
      email,
      oven_type
)
VALUES
(
      '$appliance_count',
      '$email',
      '$oven_type'
);

-- Use the same appliance_id to create the
Oven_Heat_Source
INSERT INTO Oven_Heat_Source
(
      appliance_id,
      email,
      heat_source
)
VALUES
(
      '$appliance_count',
      '$email',
      '$oven_heat_source'
);
```

   c. Jump to step 1i.

2. Else if *oven* is checked and *oven heat source* is not selected
   a. Display error message indicating that *oven heat source* for the oven must be selected.

b. Jump to step 1.d.iii.

3. If *cooktop* is checked and *cooktop heat source* is selected,
   a. Increment '$appliance_count' by 1
   b. Run the **Add Appliance Subtype** task:

```
-- Create the Cooker
INSERT INTO Cooker
(
      appliance_id,
      email,
      model_name,
      manufacturer_name
)
VALUES
(
      '$appliance_count',
      '$email',
      '$model_name',
      '$manufacturer_name'
);

-- Create the Cooktop with the same appliance_id
INSERT INTO Cooktop
(
      appliance_id,
      email,
      heat_source
)
VALUES
(
      '$appliance_count',
      '$email',
      '$cooktop_heat_source'
);
```

   c. Jump to step 1i.
4. Else if *cooktop* is checked and *cooktop heat source* is not selected
   a. Display error message indicating that *cooktop heat source* for the cooktop must be selected.
   b. Jump to step 1.d.iv.

f. Else if '$appliance_type' == 'TV':
   i. Display **Add Appliance – TV Form**.
   ii. While ***Add*** button is not clicked, do nothing.
   iii. While the *display type* is not selected and *display size* input field is not filled and *maximum resolution* is not selected:

1. *display type* ('$display_type') is selected from dropdown
2. *display size* ('$display_size') input field is filled
3. *max resolution* ('$max_resolution') is selected from dropdown.
4. If the *display type* is selected and *display size* input field is filled and *maximum resolution* is selected:
    a. Jump to step 1.f.iv.
5. Else
    a. Display error message indicating that all *display type* and *display size* and *max resolution* inputs must be filled.
    b. Jump to step 1.e.iii.

iv. If ***Add*** button is clicked, do the following:
1. Increment '$appliance_count' by 1
2. Run the **Add Appliance Subtype** task:
    a.

```
-- Create the TV
INSERT INTO TV
(     appliance_id,
      email,
      model_name,
      manufacturer_name,
      display_type,
      display_size,
      max_resolution
)
VALUES
(
      '$appliance_count',
      '$email',
      '$model_name',
      '$manufacturer_name',
      '$display_type',
      '$display_size',
      '$max_resolution'
);
```

    b. Jump to step 1i.

g. Else if '$appliance_type' == 'Washer':
i. Display **Add Appliance – Washer Form**.
ii. While ***Add*** button is not clicked, do nothing.
iii. While the *loading type* is not selected from the dropdown of option types:

1. User selects *loading type* ('$loading_type') from dropdown
2. If the *loading type* is selected:
    a. Jump to step 1.f.iv.
3. Else
    a. Display error message indicating that a *loading type* must be selected.
    b. Jump to step 1.f.iii.

iv. If ***Add*** button is clicked, do the following:
1. Increment '$appliance_count' by 1
2. Run the **Add Appliance Subtype** task:

a.
```
-- Create the Washer
INSERT INTO Washer
(
      appliance_id,
      email,
      model_name,
      manufacturer_name,
      loading_type
)
VALUES
(
      '$appliance_count',
      '$email',
      '$model_name',
      '$manufacturer_name',
      '$loading_type'
);
```

b. Jump to step 1i.

h. Else if '$appliance_type' == 'Dryer':
i. Display **Add Appliance – Dryer Form**.
ii. While ***Add*** button is not clicked, do nothing.
iii. While the *dryer heat source* is not selected from the dropdown of option types:
1. User selects *dryer heat source* ('$dryer_heat_source') from dropdown
2. If the *dryer heat source* is selected:
    a. Jump to step 1.g.iv.
3. Else
    a. Display error message indicating that a *dryer heat source* must be selected.
    b. Jump to step 1.g.iii.

iv. When ***Add*** button is clicked, do the following:
1. Increment '$appliance_count' by 1
2. Run the **Add Appliance Subtype** task:

```
-- Create the Dryer
INSERT INTO Dryer
(
      appliance_id,
      email,
      model_name,
      manufacturer_name,
      heat_source
)
VALUES
(
      '$appliance_count',
      '$email',
      '$model_name',
      '$manufacturer_name',
      '$heat_source'
);
```

a.

3. Jump to step 1i.

i. Else if '$appliance_type' == 'Refrigerator/Freezer':

i. Display **Add Appliance – Refrigerator/Freezer Form**.

ii. While ***Add*** button is not clicked, do nothing.

iii. While the *refrigerator type* is not selected from the dropdown of option types:

1. User selects *refrigerator type* ('$refrigerator_type') from dropdown
2. If the *refrigerator type* is selected:
   a. Jump to step 1.h.iv.
3. Else
   a. Display error message indicating that a *refrigerator type* must be selected.
   b. Jump to step 1.h.iv.

iv. When ***Add*** button is clicked, do the following:

1. Increment '$appliance_count' by 1
2. Run the **Add Appliance Subtype** task:

```
-- Create the Refrigerator/Freezer

INSERT INTO Refrigerator_Freezer
(
      appliance_id,
      email,
      model_name,
      manufacturer_name,
      refrigerator_type
)
VALUES
(
      '$appliance_count',
      '$email',
      '$model_name',
      '$manufacturer_name',
      '$refrigerator_type'
);
```

a.

3. Jump to step 1i.

j. Run the **Appliances Listing** task:

i. For each appliance in the household, list the '$appliance_count', the appliance type, manufacturer name, and model name:

1.

```
-- Appliance Listing

SELECT
      o.appliance_id as "Appliance #",
      "Cooker" as "Type",
      manufacturer_name as "Manufacturer",
      model_name as "Model"
FROM  Oven
JOIN  Cooker as c
ON    o.appliance_id = c.appliance_id
AND   o. email='$email'

UNION

SELECT
      ct.appliance_id as "Appliance #",
      "Cooker" as "Type",
      manufacturer_name as "Manufacturer",
      model_name as "Model"
FROM  Cooktop as ct
JOIN  Cooker as c
ON    ct.appliance_id = c.appliance_id
AND   ct.email ='$email'

UNION

SELECT
      appliance_id as "Appliance #",
      "TV" as "Type",
      manufacturer_name as "Manufacturer",
      model_name as "Model"
FROM  TV
WHERE email='$email'

UNION

SELECT
      appliance_id as "Appliance #",
      "Washer" as "Type",
      manufacturer_name as "Manufacturer",
      model_name as "Model"
FROM  Washer
WHERE email='$email'

UNION
SELECT
      appliance_id as "Appliance #",
      "Dryer" as "Type",
      manufacturer_name as "Manufacturer",
      model_name as "Model"
FROM  Dryer
WHERE email='$email'
-- cont. on next page...
```

```
UNION

SELECT
      appliance_id as "Appliance #",
      "Refrigerator/Freezer" as "Type",
      manufacturer_name as "Manufacturer",
      model_name as "Model"
FROM  Refrigerator_Freezer
WHERE email='$email';
```

ii. Display a button to add another appliance, and a ***Next*** button.

iii. If the ***Add another appliance*** button is clicked,

1. Display the **Add Appliance Form**.

iv. Else if the ***Next*** button is clicked,

1. Go to the Submission Complete page.
2. If the ***Return to main menu*** button is clicked.
    a. Go to the **Main Menu** page.

2. If the user closes the Hemkraft website or the Hemkraft website unexpectedly crashes before an appliance is added:
    a. Clear all partial data for the household;

```
-- Clear partial data in Household

DELETE FROM Phone_Number
WHERE email='$email';

DELETE FROM Household
WHERE email='$email';
```

Reports

Top 25 Popular Manufacturers

Abstract Code

1. User selects ***Top 25 Popular Manufacturers*** on the **View Reports** form.
2. Run the **List Top 25 Manufacturers** task
    a. For each manufacturer name (manufacturer) from the Manufacturer entity, count the number of appliances where the manufacturer name of the appliance is the same manufacturer name.

```
SELECT manufacturer_name as manufacturer,
  (
    SELECT COUNT(*)
    FROM Cooker as c
    WHERE manufacturer_name = manufacturer
  ) + (
    SELECT COUNT(*)
    FROM TV as t
    WHERE t.manufacturer_name = manufacturer
  ) + (
    SELECT COUNT(*)
    FROM Refrigerator_Freezer as r
    WHERE r.manufacturer_name = manufacturer
  ) + (
    SELECT COUNT(*)
    FROM Washer as w
    WHERE w.manufacturer_name = manufacturer
  ) + (
    SELECT COUNT(*)
    FROM Dryer as d
    WHERE d.manufacturer_name = manufacturer
  ) as number_of_appliances
FROM manufacturer
ORDER BY number_of_appliances DESC
LIMIT 25;
```

        i. Link is provided for drilldown report
    b. Else if the length of the list is 0
        i. Display "There are no appliances to generate the Top 25 Manufacturers"
3. If the ***View Manufacturer Drilldown Report*** is clicked for a manufacturer name ('$manufacturer_name')
    a. Run **View Manufacturer Drilldown Report** task
        i. Find the manufacturer from the Manufacturer entity whose manufacturer name == '$manufacturer_name'

ii. For each appliance type in the Refrigerator_Freezer, Cooker, Washer, Dryer, and/or TV entities and count the number of appliances from the *appliance type* associated with the '$manufacturer_name' as an integer:

```
SELECT
      "Cooker" as "Type",
      count(*) as "Raw Count"
FROM  Cooker as c
WHERE c.manufacturer_name = '$manufacturer_name'

UNION

SELECT
      "TV" as "Type",
      count(*) as "Raw Count"
FROM  TV as t
WHERE t.manufacturer_name = '$manufacturer_name'

UNION

SELECT
      "Washer" as "Type",
      count(*) as "Raw Count"
FROM  Washer as w
WHERE w.manufacturer_name = '$manufacturer_name'

UNION

SELECT
      "Dryer" as "Type",
      count(*) as "Raw Count"
FROM  Dryer as d
WHERE d.manufacturer_name = '$manufacturer_name'

UNION

SELECT
      "Refrigerator/Freezer" as "Type",
      count(*) as "Raw Count"
FROM  Refrigerator_Freezer as rf
WHERE rf.manufacturer_name = '$manufacturer_name';
```

iii. If the count for all appliance types is 0
   1. Display a message stating "There are no appliances associated with that manufacturer name"

iv. Else
   1. Display the results

Manufacturer/Model Search

Abstract Code

1. User selects ***Manufacturer/Model Search*** on the **View Reports** form.
2. While the *search* term (*'$search'*) input is not filled
    a. User enters a *search term* (*'$search'*) into the input field
    b. If the *search term* input field is filled
        i. Jump to step 3.
    c. Else
        i. Jump to step 2.
3. While the ***Submit*** button is not clicked, do nothing.
4. If the ***Submit*** button is clicked:
    a. Run the **Search** task, using the '$search'
        i. Find the set of all distinct manufacturer name(s) (manufacturer_name) from the Manufacturer entity where manufacturer_name contains the '$search' and all model names where the appliance's manufacturer == manufacturer_name UNION the set of all distinct model names (model_name) from each of the Appliance subtype entities (Cooker, TV, Refrigerator_Freezer, Washer, Dryer) where the model_name contains the '$search' and the manufacturer_name for the appliance. Sort the set by manufacturer_name ascending and model_name ascending:

```
SELECT
      manufacturer_name,
      IFNULL(model_name, '') as model
FROM Cooker
WHERE LOWER(manufacturer_name) LIKE LOWER('%$search%')
OR    LOWER(model_name) LIKE LOWER('%$search%')

UNION

SELECT
      manufacturer_name,
      IFNULL(model_name, '')
FROM  TV
WHERE LOWER(manufacturer_name) LIKE LOWER('%$search%')
OR    LOWER(model_name) LIKE LOWER('%$search%')

UNION

SELECT
      manufacturer_name,
      IFNULL(model_name, '') as model
FROM  Refrigerator_Freezer
WHERE LOWER(manufacturer_name) LIKE LOWER('%$search%')
OR    LOWER(model_name) LIKE LOWER('%$search%')

UNION

SELECT
      manufacturer_name,
      IFNULL(model_name, '') as model
FROM Washer
WHERE LOWER(manufacturer_name) LIKE LOWER('%$search%')
OR    LOWER(model_name) LIKE LOWER('%$search%')

UNION

SELECT
      manufacturer_name,
      IFNULL(model_name, '') as model
FROM Dryer
WHERE LOWER(manufacturer_name) LIKE LOWER('%$search%')
OR    LOWER(model_name) LIKE LOWER('%$search%')
ORDER BY
      manufacturer_name ASC,
      model ASC;
```

ii. If the length of the set is greater than 0:
    1. Jump to step 4b.
iii. Else
    1. Display "There were no Manufacturers or Model Names that matched your search."

b. For each manufacturer name and model name in the results of the **Search** task:
i. If the model name is NULL:
    1. Replace the model name in the result with an empty string
ii. If both the manufacturer name and model name contain the '$search' term:
    1. Highlight both cells in green
iii. Else if the manufacturer name contains the '$search' term:
    1. Highlight the cell in green
iv. Else if the model name contains the '$search' term:
    1. Highlight the cell in green
v. Display the results

Calculate Average TV Display Size by State

Abstract Code

1. User selects the ***Average TV display size by State*** report on the **View Reports** form.
2. Run the **List Average TV Display Size by State** task:
   a. Find the Location.state and average TV.display_size rounded up to the tenths decimal point from the Location, TV, Household entities from the set of all households where the postal code for the household is in the Location.state and the TV is an appliance associated with the household and sort by Location.state ascending.
   b. A link is provided for the Drilldown Report by State for each state

```
SELECT

      Location.state AS states,

   ROUND(AVG(TV.display_size), 1) AS avg_display_size

FROM TV

   INNER JOIN Household

      ON TV.email = Household.email

   INNER JOIN Location

      ON Location.postal_code = Household.postal_code

GROUP BY states

ORDER BY states ASC;
```

3. If the ***Average TV Display Report by State Drilldown Report*** is clicked for a state ('$state'):
   a. Display the '$state', TV.display_type, TV.max_resolution, and average TV.display_size as '$avg_display_size' rounded up to the tenths decimal point from Location, TV, Household entities from the set of all households where the postal code for the household is in the Location. State as the Location.state == '$state' and the TV is an appliance associated with the household and group the results by '$state', TV.display_type, TV.max_resolution, and sort by '$avg_display_size' descending.

```
SELECT
   Location.state AS states,
   TV.display_type AS display_type,
   TV.max_resolution AS max_resolution,
   ROUND(AVG(TV.display_size),1) AS avg_display_size
FROM TV
   INNER JOIN Household
      ON TV.email = Household.email
   INNER JOIN Location
      ON Location.postal_code = Household.postal_code
WHERE Location.state = '$state'
GROUP BY states, display_type, max_resolution
ORDER BY avg_display_size DESC;
```

Extra Fridge/Freezer Report

Abstract Code

1. User selects the ***List Households with Extra Fridge/Freezer*** report on the **View Reports** form.
2. Run the **Count Households with Extra Fridge/Freezer** task:
    a. Run **Count Households with Extra Fridge/Freeze** task:
        i. Display the count of the number of households from the Household entity where the count of Refrigerator_Freezer appliances that are associated with the household is greater than 1.

```
SELECT COUNT(*) AS household_count_with_extra_fridge

FROM (

      SELECT Refrigerator_Freezer.email, COUNT(*) AS
      fridge_count

      FROM Refrigerator_Freezer

      GROUP BY Refrigerator_Freezer.email

      HAVING fridge_count > 1

) AS Fridge_Count_Table;
```

    b. Run **Calculate Percentage for Households with Extra Fridge/Freeze** task:
        i. Find the Location.state and the count of the number of households from the Household entity as ('$household_count'), where the household's postal code is in Location.state and count of Refrigerator_Freezer appliances that are associated with the household is greater than 1. Sort by '$household_count' descending.
        ii. For each state in the results
            1. Count the number of households where Refrigerator_Freezer.refrigerator_type == 'chest freezer' as '$chest_freezer_count', the number of households Refrigerator_Freezer.refrigerator_type == 'upright freezer' ('$upright_freezer_count'), and the number of households Refrigerator_Freezer.refrigerator_type != 'chest

```
        freezer' and
        Refrigerator_Freezer.refrigerator_type !=
        'upright freezer' as ('$other_count')
    2. Calculate the percentage of households with each
       type of fridge/freezer as
       ('$chestFreezerPercent'),
       ('$uprightFreezerPercent'), and
       ('$otherPercent'). Round all values to whole
       number.
iii. Display the first 10 records in the results
```

```
SELECT
      hhc.state,
      COUNT(hhc.email) AS household_count,
      FORMAT((COUNT(cfc.email)/COUNT(hhc.email))*100, 'P0')
      AS chest_freezer_percent,
      FORMAT((COUNT(ufc.email)/COUNT(hhc.email))*100, 'P0')
      AS upright_freezer_percent,
      FORMAT((COUNT(oc.email)/COUNT(hhc.email))*100, 'P0') AS
      other_percent
FROM
(
      SELECT l.state, ff.email, COUNT(*) AS fridge_count
      FROM Refrigerator_Freezer ff
      INNER JOIN Household h ON ff.email = h.email
      INNER JOIN Location l ON l.postal_code = h.postal_code
      GROUP BY ff.email
      HAVING fridge_count > 1
) AS hhc
LEFT JOIN
(
      SELECT l.state, ff.email,
      COUNT(*) AS fridge_count
      FROM Refrigerator_Freezer ff
      INNER JOIN Household h ON ff.email = h.email
      INNER JOIN Location l ON l.postal_code = h.postal_code
      WHERE ff.refrigerator_type = 'chest freezer'
      GROUP BY ff.email
) AS cfc
ON hhc.email = cfc.email
LEFT JOIN
(
      SELECT l.state, ff.email, COUNT(*) AS fridge_count
      FROM Refrigerator_Freezer ff
      INNER JOIN Household h ON ff.email = h.email
      INNER JOIN Location l ON l.postal_code = h.postal_code
      WHERE ff.refrigerator_type = 'upright freezer'
      GROUP BY ff.email
) AS ufc
ON hhc.email = ufc.email
LEFT JOIN
(
      SELECT l.state, ff.email, COUNT(*) AS fridge_count
      FROM Refrigerator_Freezer ff
      INNER JOIN Household h ON ff.email = h.email
      INNER JOIN Location l ON l.postal_code = h.postal_code
      WHERE ff.refrigerator_type != 'chest freezer'
      AND ff.refrigerator_type != 'upright freezer'
      GROUP BY ff.email
) AS oc
ON hhc.email = oc.email
GROUP BY hhc.state
ORDER BY household_count DESC
LIMIT 10;
```

Laundry Center Report

Abstract Code

1. User clicks on the button for ***Laundry Center*** after choosing ***View Reports***
2. Run the **Get Washer Type and Heat Source** task:
    a. Find the most common Washer.loading_type and Dryer.heat_source) from all households from the Household entity for each state in the Location entity where the household has a Washer surrogate, and the household has a Dryer surrogate and the household's postal code is in the Location.state.
    b. Sort by Location.state ascending.

```
SELECT DISTINCT Location.state as States,
(
      SELECT
            Washer.loading_type
      FROM
            Household
            LEFT JOIN Washer
                  ON Washer.email = Household.email
            LEFT JOIN Location
                  ON Household.postal_code = Location.postal_code
      WHERE Location.state = States
      GROUP BY Washer.loading_type
      ORDER BY COUNT(Washer.loading_type) DESC
      LIMIT 1
) as "Loading Type",
(
      SELECT
            Dryer.heat_source
      FROM
            Household
            LEFT JOIN Dryer
                  ON Dryer.email = Household.email
            LEFT JOIN Location
                  ON Household.postal_code = Location.postal_code
      WHERE Location.state = States
      GROUP BY Dryer.heat_source
      ORDER BY COUNT(Dryer.heat_source) DESC
      LIMIT 1
) as "Heat Source"
FROM Household
      LEFT JOIN Location
            ON Household.postal_code = Location.postal_code
ORDER BY States ASC;
```

3. Run the **Get Households with no dryer** task:
    a. Display the count of all households as ('$household_count') from the Household entity where there is a Washer surrogate associated with the household, but no Dryer surrogate associated with the household for each state in the Location entity where the household's postal code is in the Location.state.
    b. Sort by '$household_count' descending.

```
SELECT
      Location.state as "State",
      COUNT(Household.email) as "Household Count"
FROM
      Household
      LEFT JOIN Location
            ON Household.postal_code = Location.postal_code
WHERE
      Household.postal_code = Location.postal_code
AND NOT EXISTS (
    SELECT email
    FROM Dryer
    WHERE Household.email = Dryer.email
  )
  AND EXISTS (
    SELECT email
    FROM Washer
    WHERE Household.email = Washer.email
  )
GROUP BY Location.state
ORDER BY "Household Count" DESC;
```

Bathroom Statistics

Abstract Code

1. User clicks on ***Bathroom Statistics*** button.
2. Run the **Compute min, max, average statistics per household** task:
    a. List the minimum, maximum, average count of all bathrooms as min, max, avg on total count of `Full`.bathroom_id and Half.bathroom_id per household as '$total_bathroom_count'. Round the average count of bathrooms to tenths decimal point.
    b. List the minimum, maximum, average count of full bathrooms as min, max, avg on total count of `Full`.bathroom_id if the bathroom id is in `Full` entity per household as '$total_full_bathroom_count'. Round the average count of full bathrooms to tenths decimal point.
    c. List the minimum, maximum, average count of half bathrooms as min, max, avg on total count of Half.bathroom_id if the bathroom id is in Half entity per household as '$total_half_bathroom_count'. Round the average count of half bathrooms to tenths decimal point.
    d. List the minimum, maximum, average count of bidets as min, max, avg on total count of `Full`.bidet_count and Half.bidet_count per household '$total_bidet_count'. Round the average count of bidets to tenths decimal point.
    e. List the minimum, maximum, average count of sinks as min, max, avg on total count of `Full`.sink_count and Half.sink_count per household as '$total_sink_count'. Round the average count of sinks to tenths decimal point.
    f. List the minimum, maximum, average count of commodes as min, max, avg on total count of `Full`.commode_count and Half.commode_count per household as '$total_commode_count'. Round the average count of commodes to tenths decimal point.
    g. List the minimum, maximum, average count of bathtubs as min, max, avg on total count of `Full`.tub_count per household as '$total_bathtub_count'. Round the average count of bathtubs to tenths decimal point.
    h. List the minimum, maximum, average count of all showers as min, max, avg on total count of `Full`.shower_count per household as '$total_shower_count'. Round the average count of showers to tenths decimal point.
    i. List the minimum, maximum, average count of tub/showers as min, max, avg on total count of `Full`.tub_shower_count per household as '$total_tubshower_count'. Round the average count of tub/showers to tenths decimal point.

```
SELECT
MAX(totalCountTable.total_bathroom_count) AS max_bathroom_count,
MIN(totalCountTable.total_bathroom_count) AS min_bathroom_count,
ROUND((SUM(totalCountTable.total_bathroom_count)/COUNT(*)),10) AS
avg_bathroom_count,
MAX(totalCountTable.total_full_bathroom_count) as max_full_bathroom_count,
MIN(totalCountTable.total_full_bathroom_count) AS min_full_bathroom_count,
ROUND((SUM(totalCountTable.total_full_bathroom_count)/COUNT(*)),10) AS
avg_full_bathroom_count,
MAX(totalCountTable.total_half_bathroom_count) AS max_half_bathroom_count,
MIN(totalCountTable.total_half_bathroom_count) AS min_half_bathroom_count,
ROUND((SUM(totalCountTable.total_half_bathroom_count)/COUNT(*)),10) AS
avg_half_bathroom_count,
MAX(totalCountTable.total_commode_count) as max_commode_count,
MIN(totalCountTable.total_commode_count) as min_commode_count,
ROUND((SUM(totalCountTable.total_commode_count)/COUNT(*)),10) as
avg_commode_count,
MAX(totalCountTable.total_sink_count) as max_sink_count,
MIN(totalCountTable.total_sink_count) as min_sink_count,
ROUND((SUM(totalCountTable.total_sink_count)/COUNT(*)),10) as avg_sink_count,
MAX(totalCountTable.total_bidet_count) as max_bidet_count,
MIN(totalCountTable.total_bidet_count) as min_bidet_count,
ROUND((SUM(totalCountTable.total_bidet_count)/COUNT(*)),10) as avg_bidet_count,
MAX(totalCountTable.total_bathtub_count) as max_bathtub_count,
MIN(totalCountTable.total_bathtub_count) as min_bathtub_count,
ROUND((SUM(totalCountTable.total_bathtub_count)/COUNT(*)),10) as
avg_bathtub_count,
MAX(totalCountTable.total_shower_count) as max_shower_count,
MIN(totalCountTable.total_shower_count) as min_shower_count,
ROUND((SUM(totalCountTable.total_shower_count)/COUNT(*)),10) as
avg_shower_count,
MAX(totalCountTable.total_tub_shower_count) as max_tub_shower_count,
MIN(totalCountTable.total_tub_shower_count) as min_tub_shower_count,
ROUND((SUM(totalCountTable.total_tub_shower_count)/COUNT(*)),10) as
avg_tub_shower_count
FROM
(
SELECT email as email, SUM(derivedTable.bathroom_count) AS
total_bathroom_count,
SUM(derivedTable.full_bathroom_count) as total_full_bathroom_count,
SUM(derivedTable.half_bathroom_count) as total_half_bathroom_count,
SUM(derivedTable.bathroom_commode_count) as total_commode_count,
SUM(derivedTable.bathroom_sink_count) as total_sink_count,
SUM(derivedTable.bathroom_bidet_count) as total_bidet_count,
SUM(derivedTable.bathroom_bathtub_count) as total_bathtub_count,
SUM(derivedTable.bathroom_shower_count) as total_shower_count,
SUM(derivedTable.bathroom_tub_shower_count) as total_tub_shower_count
FROM
(
    SELECT  email as "email", COUNT(*) as bathroom_count,
    COUNT(*) as full_bathroom_count, NULL as half_bathroom_count,
    SUM(commode_count) as bathroom_commode_count,
    SUM(sink_count) as bathroom_sink_count,
    SUM(bidet_count) as bathroom_bidet_count,
    SUM(bathtub_count) as bathroom_bathtub_count,
    SUM(shower_count) as bathroom_shower_count,
    SUM(tub_shower_count) as bathroom_tub_shower_count
      FROM `Full` GROUP BY email
    UNION ALL
    SELECT email as "email", COUNT(*) as bathroom_count,
    NULL as full_bathroom_count, COUNT(*) as half_bathroom_count,
    SUM(commode_count) as bathroom_commode_count,
    SUM(sink_count) as bathroom_sink_count,
    SUM(bidet_count) as bathroom_bidet_count,
    NULL as bathroom_bathtub_count,
    NULL as bathroom_shower_count,
    NULL as bathroom_tub_shower_count
    FROM Half GROUP BY email
) derivedTable
GROUP BY email) totalCountTable;
```

3. Run **Compute the total bidet count per state** task:
   a. For each distinct Location.state in the Location entity, count the total `Full`.bidet_count and Half.bidet_count as 'total_bidet_count_per_state' from all bathrooms from all households from the Household entity where the household's postal code is in the Location.state
   b. Group by Location.state.
   c. Sort the results based on 'total_bidet_count_per_state' in descending order.
   d. Display the Location.state and '$total_bidet_count_per_state' of the first record.

```
SELECT totalBidetCountPerStateTable.state as state,
totalBidetCountPerStateTable.total_bidet_count_state as
max_bidet_count_state
FROM
(
SELECT Location.state as state,
SUM(totalBidetCountPerPostalCode.total_bidet_count_postal_code) as
total_bidet_count_state
FROM
(SELECT joinedTable.postal_code as postal_code,
SUM(joinedTable.total_bidet_count) as total_bidet_count_postal_code
FROM
(SELECT Household.postal_code as postal_code, derivedTable.email as
email ,
SUM(derivedTable.bathroom_bidet_count) as total_bidet_count
FROM
(
    SELECT  email as "email", SUM(bidet_count) as
bathroom_bidet_count
      FROM `Full` GROUP BY email
    UNION ALL
    SELECT email as "email", SUM(bidet_count) as
bathroom_bidet_count
      FROM Half GROUP BY email
) derivedTable
INNER JOIN Household ON Household.email = derivedTable.email
GROUP BY email) joinedTable
GROUP BY postal_code ) totalBidetCountPerPostalCode
INNER JOIN Location ON Location.postal_code =
totalBidetCountPerPostalCode.postal_code
GROUP BY state
)totalBidetCountPerStateTable
ORDER BY totalBidetCountPerStateTable.total_bidet_count_state DESC
LIMIT 1;
```

4. Run **Compute the total bidet count per postal Code** task:
    a. For a household, find the corresponding postal code, '$postal_code' and total count of `Full`.bidet_count and Half.bidet_count as '$total_bidet_count' computed in **Compute min, max, average statistics per household.**
    b. Group by '$postal_code' and find the total bidet count per postal code 'total_bidet_count_per_postal_code'.
    c. Sort the results based on 'total_bidet_count_per_postal_code' in descending order.
    d. Display the '$postal_code' and '$total_bidet_count' of the first record.

```
SELECT totalBidetCountPerPostalCodeTable.postal_code as
postal_code,
totalBidetCountPerPostalCodeTable.total_bidet_count_postal_code as
max_bidet_count_postal_code
FROM
(
SELECT joinedTable.postal_code as postal_code,
SUM(joinedTable.total_bidet_count) as total_bidet_count_postal_code
FROM
(SELECT Household.postal_code as postal_code, derivedTable.email as
email ,
SUM(derivedTable.bathroom_bidet_count) as total_bidet_count
FROM
(
    SELECT  email as "email", SUM(bidet_count) as
bathroom_bidet_count
      FROM `Full` GROUP BY email
    UNION ALL
    SELECT email as "email", SUM(bidet_count) as
bathroom_bidet_count
      FROM Half GROUP BY email
) derivedTable
INNER JOIN Household ON Household.email = derivedTable.email
GROUP BY email) joinedTable
GROUP BY postal_code
) totalBidetCountPerPostalCodeTable
ORDER BY
totalBidetCountPerPostalCodeTable.total_bidet_count_postal_code
DESC
LIMIT 1;
```

5. Run **Compute the count of Single primary bathroom** task:
    a. For each household, find the total count of `Full`.bathroom_id per household $single_primary_bathroom_count' where 'single_primary_bathroom_count is equal to 1 and `Full`.is_primary is true and Household.email is in `Full` entity, but not in Half entity.

```
SELECT singlePrimaryFullTable.single_primary_bathroom_count as
single_primary_bathroom_count
FROM
(SELECT email as email, count(*)  as single_primary_bathroom_count,
is_primary as is_primary from `Full`
GROUP BY email
HAVING single_primary_bathroom_count = 1 and is_primary = 1 and
email NOT IN (SELECT email from Half))singlePrimaryFullTable;
```

Household Averages by Radius

Abstract Code

1. Display **Household Averages by Radius** form.
2. While ((*postal code* ('$postal_code') is not inputted) or (*postal code* ('$postal_code') is invalid) or (*search radius* ('$search_radius') is not inputted)):
   a. *postal code* ('$postal_code') is inputted
   b. *search radius* ('$search_radius') is inputted
   c. If ***Submit*** button is clicked:
      i. If the '$postal_code' is not found in the Location entity

```
SELECT postal_code
FROM Location
WHERE postal_code = '$postal_code';
```

         1. Display an error message indicating that the *postal code* input was invalid and empty the input fields.
         2. Jump to step 2.
      ii. Else
         1. Jump to step 2.e.
   d. Run the **Household Averages by Radius** task:
   e. For each postal_code in the set of all unique Location.postal_code from the Location entity and their Location.longitude and Location.latitude
      i. Calculate the '$haversine_distance' to determine if the '$postal_code' is in the specified '$search_radius'
      ii. If the '$haversine_distance' is less than or equal to the search radius:
         1. Run the **Household Averages by Radius** task:
         2. Find all households from the Household entity where the household lives in the '$postal_code'
         3. Round the average count of all bathrooms on total count of bathrooms per household ('$average_bathroom_count') to the nearest tenths.
         4. Round the average count of all bedrooms on total count of bedrooms per household in ('$average_bedroom_count') to the nearest tenths. Display '$average_bedroom_count'.
         5. Round the average count of all occupants per household in ('$average_occupant_count') to the nearest integer. Display '$average_occupant_count'.

6. Round the division of the sum of all Household.occupant_count and the total count of Bathroom.commode_count (`Full` and Half) for each household where the bathroom is associated with the household and the household is in the '$postal_code' to determine ('$ratio_commodes_to_occupants'). Display the ratio as "1: '$ratio_commodes_to_occupants'."
7. Divide the count of all appliances from the Cooker, TV, Washer, Dryer, and Refrigerator_Freezer entities ('$total_number_of_appliances') by the count of unique Household.email ('$number_of_households') to determine '$average_number_of_appliances'. Round '$average_number_of_appliances' to the nearest tenths decimal point. Display '$average_number_of_appliances'.
8. Find all '$heat_source' on Dryer, Cooktop, and Oven entities where the appliance is associated with the household and the household lives in the '$postal_code'
    a. Display the most common '$heat_source'.

```
SELECT

      ROUND(SUM(bedroom_count)/COUNT(email), 1) as avg_bedroom_count,
      ROUND(SUM(ct)/COUNT(email), 1) as avg_bathroom_count,
      CEIL(SUM(occupant_count)/COUNT(email)) as avg_occupant_count,
      CONCAT("1:", ROUND(SUM(occupant_count)/SUM(commode_count), 2)) as
      ratio_commodes_to_occupants,
      ROUND(SUM(ap)/COUNT(email), 1) as avg_appliance_count,
     heat_source as most_common_heat_source

FROM
(
      SELECT COUNT(*) as ct, COUNT(commode_count) as commode_count
      FROM Half
      UNION
      SELECT COUNT(*) as ct, COUNT(commode_count) as commode_count
      FROM `Full`
) as sum_bathrooms,
(
      SELECT COUNT(*) as ap
      FROM Cooker
      UNION
      SELECT COUNT(*) as ap
      FROM TV
```

```
      UNION
      SELECT COUNT(*) as ap
      FROM Refrigerator_Freezer
      UNION
      SELECT COUNT(*) as ap
      FROM Washer
      UNION
      SELECT COUNT(*) as ap
      FROM Dryer
) as sum_appliances,
(
      SELECT COUNT(heat_source) as heat_source_count, heat_source
      FROM Dryer
      UNION
      SELECT COUNT(heat_source) as heat_source_count, heat_source
      FROM Cooktop
      UNION
      SELECT COUNT(heat_source) as heat_source_count, heat_source
      FROM Oven_Heat_Source
      ORDER BY heat_source_count DESC
      LIMIT 1
) as heat_sources,
Household
WHERE Household.postal_code in (
      SELECT house_postal_code2
      FROM
      (
            SELECT house_postal_code1, house_postal_code2,
                        3958.75*2*atan2(sqrt(sin((Radians(latitude2) -
            Radians(latitude1))/2)*sin((Radians(latitude2) -
            Radians(latitude1))/2) +
            cos(Radians(latitude1))*cos(Radians(latitude2))*sin((Radians(
            longitude2)-Radians(longitude1))/2)*sin((Radians(longitude2)-
            Radians(longitude1))/2)), sqrt(1-sin((Radians(latitude2) -
            Radians(latitude1))/2)*sin((Radians(latitude2) -
            Radians(latitude1))/2) +
            cos(Radians(latitude1))*cos(Radians(latitude2))*sin((Radians(
            longitude2)-Radians(longitude1))/2)*sin((Radians(longitude2)-
            Radians(longitude1))/2))) as haversine_distance
            FROM
            (
                  SELECT DISTINCT h.postal_code as house_postal_code1,
                  l.latitude as latitude1, l.longitude as longitude1
                  FROM Household h
                  INNER JOIN Location l ON h.postal_code = l.postal_code
                  WHERE h.postal_code  = '$postal_code'
            ) as a
            CROSS JOIN
            (
                  SELECT DISTINCT h.postal_code as house_postal_code2,
                  l.latitude as latitude2 , l.longitude as longitude2
```

```
                FROM Household h
                INNER JOIN Location l ON h.postal_code = l.postal_code
            ) as b HAVING haversine_distance <= '$radius') as c

    );
```