CS6400 -- Phase 2 Abstract Code + SQL (Team 101)

Manufacturers

- Run an "INSERT" query to insert values into the `manufacturer` table: This is assuming that we do have access to the list of all manufactures of appliances, and are inserting the information in a local table; (0)

Queries

(0) Manufacturer tables; subject to change only externally

```
INSERT INTO manufacturer (manufacturer_name) VALUES
(MANUFACTURER_1),
(MANUFACTURER_2),
(MANUFACTURER_3),
...
(MANUFACTURER_N);
```

Email Form

Abstract Code

- User enters email (`$Email`) in input field

- When Submit button is pressed:
  - Run "SELECT" query on database to get all rows from the main household table where email = `$Email` (1)
  - If Email is already in database (i.e. if query > 0):
  - Go back to Email Form with error message
  - Else (if query = 0):
  - Proceed to Postal Code Search

Queries

(1) Select on Email table

```
SELECT COUNT(email)
FROM household
WHERE email = '$Email'
```

Postal Code Search

Abstract Code

- User enters postal code (`$Postal_Code`) in input field
  - When Submit button is pressed:
    - Run "SELECT" query on database to get row from the city_info table where postal code = `$Postal_Code` (2)
    - If postal code is in database of postal codes (i.e. if query returns 1 row):
      - Display Query Result (2)
      - Ask User if the information is correct
      - When Yes button is selected:
        - Proceed to Phone Number Form
      - When No button is selected:
        - Return to beginning of Postal Code Form
    - Else (i.e. if query returns 0 rows):
      - return Postal Code Form with error message

Queries

(2) Select on Postal Code table

```
SELECT postal_code, city, state
FROM city_info
WHERE postal_code = '$Postal_Code'
```

Phone Number Form

Abstract Code

- Ask user if they would like to enter a phone number
- If Yes button is selected:
  - Display Area Code, Number, and Phone type text entries fields
  - User enters area code (`$Area_Code`) in input field
  - User enters remaining 7 digits (`$7_Digits`) in input field
  - User Selects phone type (`$Phone_Type`) from drop down list of: home, mobile, work or other
  - Run a "SELECT" statement on the phone_info table to check if the phone number is already in the database (3)
  - If there is a match (i.e. if query > 0):
    - Show error message
  - Else (i.e. if query = 0):
    - Run an "INSERT" statement on the phone_info table to add the phone number to the database (4)
    - When Next button is selected:
      - Proceed to Household Info Form
- If No button is selected:
  - Proceed to Household Info Form

Queries

(3) Select on Phone Info table

```
    SELECT COUNT(phone_number)
    FROM phone_info
    WHERE area_code = '$Area_Code'
    AND phone_number = '$7_Digits'
```

(4) Insert into Phone Info table

```
    INSERT INTO phone_info
    (area_code, phone_number, phone_type, email)
    VALUES ('$Area_Code', '$7_Digits', '$Phone_Type', '$Email')
```

Household Info Form

Abstract Code

- User selects home type (`$Home_Type`) from a drop down list of: house, apartment, townhome, condominium, or mobile home
- User enters square footage (`$Square_Footage`) in input field
- User enters number of occupants (`$Number_Of_Occupants`) in input field
- User enters number of bedrooms (`$Number_Of_Bedrooms`) in input field
- When Next button is selected:
    - Run an "INSERT" statement on the household table to add the household information to the database (5)
    - Proceed to Bathroom Info Form

Queries

(5) Insert into Household table

```
    INSERT INTO household
    (email, household_type, square_footage, number_of_occupants, number_of_bedrooms, p
ostal_code
    VALUES (
        '$Email',
        '$Home_Type',
        '$Square_Footage',
        '$Number_Of_Occupants',
        '$Number_Of_Bedrooms',
        '$Postal_Code'*
    )
```

Bathroom Info Form

Abstract Code

- Initialize `$Bathroom_#` = 0
- Ask user if they have a half bathroom
- If Yes is selected (tab selecting "half"):
    - User enters sinks (`$Number_of_Sinks`) in input field
    - User enters commodes (`$Number_of_Commodes`) in input field
    - User enters bidets (`$Number_of_Bidets`) in input field
    - User enters half bath name (`$Half_Bath_Name`) in input field (This field is optional)
    - When Add is selected:
        - Increment bathroom # (`$Bathroom_#`) by 1
        - Run an "INSERT" statement on the half bathroom table to add the bathroom information to the database (6)
        - Jump to bathroom listing
- If No is selected (tab selecting "full"):
    - User enters sinks (`$Number_of_Sinks`) in input field
    - User enters commodes (`$Number_of_Commodes`) in input field
    - User enters bidets (`$Number_of_Bidets`) in input field
    - User enters bathtubs (`$Number_of_Bathtubs`) in input field
    - User enters showers (`$Number_of_Showers`) in input field
    - User enters tub/shower (`$Number_of_TubShowers`) in input field
    - Run "SELECT" statement on the full bathroom table to check if any of the already entered bathroom is marked as primary (7)
    - If primary bathroom variable (`$Is_Primary_Bathroom`) is not yet entered (i.e. if query = false):

        - User may select the "is primary bathroom" (`$Is_Primary_Bathroom`) check box
    - Else: (i.e. query = true)
        - User can not select check box
    - When Add is selected:
        - Increment bathroom # (`$Bathroom_#`) by 1
        - Run an "INSERT" statement on the full bathroom table to add the bathroom information to the database (8)
        - Jump to bathroom listing
- If No is selected (Cannot happen if user interface is used correctly):
    - Display an error message indicating that the user must have one bathroom
    - Return to Bathroom Info Form

Queries

(6) Insert into Half Bathroom table

```
    INSERT INTO half_bathroom
    (email, bathroom_number, number_of_sinks, number_of_commodes, number_of_bidets, ha
lf_bath_name)
    VALUES (
        '$Email',
        '$Bathroom_#',
        '$Number_of_Sinks',
        '$Number_of_Commodes',
        '$Number_of_Bidets',
        '$Half_Bath_Name'
    )
```

(7) Select from Full Bathroom table

```
    SELECT EXISTS(
    SELECT email, bathroom_number
    FROM full_bathroom
    WHERE email = '$Email'
    AND is_primary_bathroom = true
    )
```

(8) Insert into Full Bathroom table

```
    INSERT INTO full_bathroom
    (email, bathroom_number, number_of_sinks, number_of_commodes, number_of_bidets, nu
mber_of_bathtubs, number_of_showers, number_of_tub_showers, is_primary_bathroom)
    VALUES (
        '$Email',
        '$Bathroom_#',
        '$Number_of_Sinks',
        '$Number_of_Commodes',
        '$Number_of_Bidets',
        '$Number_of_Bathtubs',
        '$Number_of_Showers',
        '$Number_of_TubShowers',
        '$Is_Primary_Bathroom'
    )
```

Bathroom Listing

Abstract Code

- Run "SELECT" statement on the both bathroom tables to get all bathrooms associated with the email (9) (Since this will be shown after each entry of households, there must exist at least one data to display.)
- User responds if they want to Add another bathroom
- When Add Another Bathroom is selected:
    - Return to the beginning of Bathroom Form
- When Next is selected:
    - Continue to Appliance Form

Queries

(9) Select from Full Bathroom table

```
SELECT bathroom_number, bathroom_type, is_primary_bathroom FROM (
    SELECT bathroom_number, 'half' AS bathroom_type, false AS is_primary_bathroom
    FROM half_bathroom
    WHERE email = '$Email'
    UNION
    SELECT bathroom_number, 'full' AS bathroom_type, is_primary_bathroom AS is_primary
_bathroom
    FROM full_bathroom
    WHERE email = '$Email'
```

```
) AS bathroom_listing
ORDER BY bathroom_number
```

Appliances Info Form

Abstract Code

- Initialize `$Appliance_#` = 0
- User selects appliance type (`$Appliance_Type`) from a drop down list of: dryer, fridge, TV, washer, and - cooker.
- User selects manufacturer (`$Manufacturer`) from a drop down list of manufacturers.
- User enters model (`$Model`) in input field (This field is optional)
- If appliance type Dryer is selected:
  - set `$Dryer` = true
  - User selects heat source (`$Heat_Source`) from a drop down list of: gas, electric, or none
- If appliance type Fridge is selected:
  - set `$Fridge` = true
  - User selects fridge type (`$Fridge_Type`) from a drop down list of: bottom freezer refrigerator, French door refrigerator, side-by-side refrigerator, top freezer refrigerator, chest freezer, or upright freezer.
- If appliance type Washer is selected:
  - set `$Washer` = true
  - User selects loading type (`$Loading_Type`) from a drop down list of: top or front
- If appliance type TV is selected:
  - set `$TV` = true
  - User selects display type (`$Display_Type`) from a drop down list of: tube, DLP, plasma, LCD, or LED
  - User selects max resolution (`$Max_Resolution`) from a drop down list of: 480i, 576i, 720p, 1080i, 1080p, - 1440p, 2160p (4K), or 4320p (8K)

    - User enters display size(`$Display_Size`) in input field
- If appliance type Cooker is selected:
    - User selects Oven or Cooktop checkbox
    - If Cooktop is checked:
        - set `$Cooktop` = true
        - User selects cook heat source (`$Cook_Heat_Source`) from a drop down list of: gas, electric, radiant electric, or induction
    - If Oven is checked:
        - set `$Oven` = true
        - User selects oven heat source (`$Oven_Heat_Source`) from a checkbox list of: gas, electric, and or microwave
        - User selects oven type (`$Oven_Type`) from a drop down list of: gas, electric, radiant electric, or induction
- When Add button is selected:
    - Increment appliance # (`$Appliance_#`) by 1
    - Run an "INSERT" statement on the appliance table to add the appliance information to the database (10)
    - if `$Dryer` = true:
        - Run an "INSERT" statement on the dryer table to add the dryer information to the database (11)
    - if `$Fridge` = true:
        - Run an "INSERT" statement on the fridge table to add the fridge information to the database (12)
    - if `$Washer` = true:
        - Run an "INSERT" statement on the washer table to add the washer information to the database (13)
    - if `$TV` = true:
        - Run an "INSERT" statement on the TV table to add the TV information to the database (14)
    - if `$Cooktop` = true:

        - Run an "INSERT" statement on the cooktop table to add the cooktop information to the database (15)
    - if `$Oven` = true:
        - Run an "INSERT" statement on the oven table to add the oven information to the database (16)
        - For each oven heat source selected:
            - Run an "INSERT" statement on the oven_heat_source table to add the oven heat source information to the database (17)
    - Jump to appliance listing

Queries

(10) Insert into Appliance table

```
INSERT INTO appliance
(email, appliance_number, appliance_type, manufacturer_id, model, type)
VALUES (
    '$Email',
    '$Appliance_#',
    '$Appliance_Type',
    '$Manufacturer',
    '$Model',
    '$Appliance_Type'
)
```

(11) Insert into Dryer table

```
INSERT INTO dryer
(email, appliance_number, heat_source)
VALUES (
    '$Email',
    '$Appliance_#',
    '$Heat_Source'
)
```

(12) Insert into Fridge table

```
INSERT INTO fridge
(email, appliance_number, fridge_type)
VALUES (
    '$Email',
```

```
    '$Appliance_#',
    '$Fridge_Type'
)
```

(13) Insert into Washer table

```
INSERT INTO washer
(email, appliance_number, loading_type)
VALUES (
    '$Email',
    '$Appliance_#',
    '$Loading_Type'
)
```

(14) Insert into TV table

```
INSERT INTO TV
(email, appliance_number, display_type, max_resolution, display_size)
VALUES (
    '$Email',
    '$Appliance_#',
    '$Display_Type',
    '$Max_Resolution',
    '$Display_Size'
)
```

(15) Insert into Cooktop table

```
INSERT INTO cooktop
(email, appliance_number, cook_heat_source)
VALUES (
    '$Email',
    '$Appliance_#',
    '$Cook_Heat_Source'
)
```

(16) Insert into Oven table

```
INSERT INTO oven
(email, appliance_number, oven_type)
VALUES (
    '$Email',
    '$Appliance_#',
    '$Oven_Type'
)
```

(17) Insert into Oven_Heat_Source table

```
    INSERT INTO oven_heat_source
    (email, appliance_number, oven_heat_source)
    VALUES (
        '$Email',
        '$Appliance_#',
        '$Oven_Heat_Source'
    )
```

Appliance Listing

Abstract Code

- Run a "SELECT" statement on the appliance table to get the appliance information from the database (18)
- Display a table of all appliances types added in the order in which they were added (appliance ID) as well the manufacturer and model
- User responds if they want to Add another appliance
- When Add Another Appliance is selected:
    - Return to the beginning of Appliance Form
- When Next is selected:
    - Continue to Done Screen

```
    INSERT INTO oven_heat_source
    (email, appliance_number, oven_heat_source)
    VALUES (
        '$Email',
        '$Appliance_#',
        '$Oven_Heat_Source'
    )
```

Queries

(18) Select from Appliance table

```
    SELECT email, appliance_number, appliance_type, manufacturer, model, type FROM app
liance
```

```
WHERE email = '$Email'
ORDER BY appliance_number ASC
```

Top 25 Popular Manufacturers Report

Abstract Code

- Show a summary of each manufacturer's total count of appliances recorded by users so far, but limited to the top 25 (No parameters are needed, and thus no use-input data validation is needed) (19)
- User selects ('$Manufacturer Name') from a drop-down list of all manufacturers recognized.
    - This must be found in the existing appliances records made so far to show any data, otherwise show a message "No data has been found for your request.".
- Show a summary for each appliance type, such as cooker, and TV, the total records made by users, for the specific ('$Manufacturer Name') specified. (20)

Queries

(19) Selecting top 25 manufacturer names and count

```
SELECT manufacturer_name, COUNT(manufacturer_name) AS total_appliance_count
FROM manufacturer NATURAL JOIN appliance
GROUP BY manufacturer_name
ORDER BY COUNT(manufacturer_name) DESC
LIMIT 25
```

(20) Select from appliances for a particular manufacturer

```
SELECT appliance_type, COUNT(*) AS total_appliance_count
FROM manufacturer
NATURAL JOIN appliance
WHERE manufacturer_name = '$Manufacturer_Name'
GROUP BY appliance_type
```

Manufacturer/Model Search Report

Abstract Code

- User enters (‘$Search Parameter’)
    - The parsed (lower-case) result of the (‘$Search Parameter’) must be found in the existing appliances model name or the manufacturer name, recognized in the database to show any data. The search should also show partial matches as well.
- Show all the recognized manufacturer name and model name pair, where the parsed (lower-case) result of the (‘$Search Parameter’) matches or partially matches with either the manufacturer name and/or the model name. (21)
    - The data would be organized so that the manufacturer name and the model name are in alphabetical order, with higher priority on manufacturer.
- Apply cell background highlight to the manufacturer name and/or model name to the items considered a match.

Queries

(21) Select from manufacturer and model

```
SELECT manufacturer_name AS Manufacturer, model AS Model
FROM manufacturer
NATURAL JOIN appliance
WHERE manufacturer_name LIKE '%Search Parameter%'
OR model LIKE '%Search Parameter%'
ORDER BY manufacturer_name ASC, model ASC
```

Average TV Display Size by State Report

Abstract Code

- Display the summary for the average tv size for each of the states. (22)
    - The average tv size would be rounded to the nearest tenths decimal which is found by taking the sum of all the tv sizes and dividing by the total number of tvs in the database.
    - The data would be organized by the name of the state in an alphabetical order.
- User selects a (‘$State’) from a drop-down of all states.
    - At least 1 tv must be registered in the selected state to show any data, otherwise show the message “No data has been found for the selected

state.”

- Display the average tv size for each screen type and maximum resolution, for the specified (‘$State’). (23)
    - Depending on the query result (0 rows), the frontend of the application will display the message “No data has been found for the selected state.”

Queries

(22) Select from TV table

```
    SELECT state, ROUND(AVG(display_size), 1) AS average_tv_size
    FROM TV
    NATURAL JOIN appliance
    NATURAL JOIN household
    NATURAL JOIN city_info
    GROUP BY state
    ORDER BY state ASC
```

(23) Select from TV table for a particular state

```
    SELECT display_type, max_resolution, ROUND(AVG(display_size), 1) AS average_tv_siz
e
    FROM TV
    NATURAL JOIN appliance
    NATURAL JOIN household
    NATURAL JOIN city_info
    WHERE state = '$State'
    GROUP BY display_type, max_resolution
    ORDER BY display_type ASC, max_resolution ASC
```

Extra Fridge/Freezer Report

Abstract Code

- Show sum of all the households that have more than 1 fridge/freezer in their house. (24)
- Show a summary of households that have more than 1 fridge/freezer in their house, for each state.
    - The data would be organized from the states with the most number of households that have more than 1 fridge/freezer, to the least. This is found by adding each state's total number of fridge or freezer.

- The output will only show the upper 10 in this ranking, and anything below would be not displayed.
- In addition to the state, and count of households with more than 1 fridge or freezer, we would also report:
  - the percentage of households with multiple fridge/freezers in that state with chest freezers. (Found by doing the total number of chest freezers divided by the total number for that state)
  - the percentage of households with multiple fridge/freezers in that state with an upright freezer. (Found by doing the total number of upright freezers divided by the total number for that state)
  - the percentage of households with multiple fridge/freezers in that state with something else. (Found by doing the total number of appliances that are not the type of chest or upright divided by the total number for that state)
    - Which all of them would be rounded to the nearest integer.

Queries

(24) Select from fridge/freezer table

```
    SELECT COUNT(*) AS households_with_multiple_fridge_freezers
    FROM fridge
    NATURAL JOIN appliance
    NATURAL JOIN household
    GROUP BY email
    HAVING COUNT(*) > 1
```

(25) Select from fridge/freezer table

```
    SELECT state, COUNT(*) AS households_with_multiple_fridge_freezers,
    ROUND((SUM(CASE WHEN fridge_type = 'chest' THEN 1 ELSE 0 END) / COUNT(*)) * 100) A
S chest_freezer_percentage,
    ROUND((SUM(CASE WHEN fridge_type = 'upright' THEN 1 ELSE 0 END) / COUNT(*)) * 100)
AS upright_freezer_percentage,
    ROUND((SUM(CASE WHEN fridge_type != 'chest' AND fridge_type != 'upright' THEN 1 EL
SE 0 END) / COUNT(*)) * 100) AS other_percentage
    FROM fridge
    NATURAL JOIN appliance
    NATURAL JOIN city_info
    GROUP BY state
    HAVING COUNT(*) > 1
```

```
    ORDER BY COUNT(*) DESC
    LIMIT 10
```

Laundry Center Report

Abstract Code

- Show most common (with the highest count values) loading type and heat source for each state (26)
    - The data would be organized by showing the states in alphabetical order.
- Show the summary of the household count with washing machine but no dryer for each state. (27)
    - The data would be organized by ordering the counts high to low. (States with the most households with washing machines, without a dryer, would come first.)

Queries

(26) Select from dryer and washer tables

NOTE: MySQL 8.0 and up is required.

```
SELECT subq_washer.state, loading_type, dryer_heat_source FROM
(
    SELECT
    state,
    loading_type,
    COUNT(*) AS total_washer_households,
    ROW_NUMBER() OVER (PARTITION BY state ORDER BY COUNT(*) DESC) AS row_number_washer
    FROM washer
    NATURAL JOIN appliance
    NATURAL JOIN household
    NATURAL JOIN city_info
    GROUP BY state, loading_type
) AS subq_washer
LEFT JOIN (
    SELECT
    state,
    dryer_heat_source,
    COUNT(*) AS total_dryer_households,
    ROW_NUMBER() OVER (PARTITION BY state ORDER BY COUNT(*) DESC) AS row_number_dryer
    FROM dryer
    NATURAL JOIN appliance
    NATURAL JOIN household
    NATURAL JOIN city_info
    GROUP BY state, dryer_heat_source
```

```
) AS subq_dryer
ON subq_washer.state = subq_dryer.state
WHERE row_number_washer = 1 AND row_number_dryer = 1
```

(27) Select from dryer and washer tables

```
SELECT state, COUNT(*) AS count_washer FROM
(
SELECT state, subq_washer.email, total_washer
FROM
(
    SELECT
    state,
    email,
    COUNT(*) AS total_washer
    FROM washer
    NATURAL JOIN appliance
    NATURAL JOIN household
    NATURAL JOIN city_info
    GROUP BY state, email
) AS subq_washer
INNER JOIN
(
  SELECT email, total_dryer
  FROM
  (
    SELECT
    email,
    COUNT(*) AS total_dryer
    FROM dryer
    NATURAL JOIN appliance
    NATURAL JOIN household
    GROUP BY email
) AS subq_subq_dryer
WHERE total_dryer = 0
) subq_dryer
ON subq_washer.email = subq_dryer.email
) AS main
GROUP BY state
ORDER BY COUNT(*) DESC
```

Bathroom Statistics Report

Abstract Code

- Show aggregate information (min, max, avg) of count of all bathrooms per household (28)
- Show aggregate information (min, max, avg) of count of half bathrooms per household (29)

- Show aggregate information (min, max, avg) of count of full bathrooms per household (30)
- Show aggregate information (min, max, avg) of count of commodes per household (31)
- Show aggregate information (min, max, avg) of count of sinks per household (32)
- Show aggregate information (min, max, avg) of count of bidet per household (33)
- Show aggregate information (min, max, avg) of count of bathtubs per household (34)
- Show aggregate information (min, max, avg) of count of showers per household (35)
- Show aggregate information (min, max, avg) of count of tub/showers per household (36)
- State with most bidet and its count (37)
- Postal code with most bidets and its count (38)
- Count of households with one primary full bathroom and none of the other. (39)

Queries

(28) Aggregate info for all bathrooms

```
SELECT
MAX(count_baths) AS max_count_all_bathroom,
MIN(count_baths) AS min_count_all_bathroom,
AVG(count_baths) AS avg_count_all_bathroom
FROM
(
SELECT
    email,
    COUNT(*) AS count_baths
FROM (
    SELECT email, bathroom_number
    FROM half_bathroom
    UNION
    SELECT email, bathroom_number
    FROM full_bathroom
) AS subq
NATURAL JOIN household
GROUP BY email
) AS main
```

(29) Aggregate info for all half bathrooms

```
SELECT
MAX(count_baths) AS max_count_half_bathroom,
MIN(count_baths) AS min_count_half_bathroom,
AVG(count_baths) AS avg_count_half_bathroom
FROM
(
SELECT
    COUNT(*) as count_baths
FROM half_bathroom
NATURAL JOIN household
GROUP BY email
) AS subq
```

(30) Aggregate info for all full bathroom

```
SELECT
MAX(count_baths) AS max_count_full_bathroom,
MIN(count_baths) AS min_count_full_bathroom,
AVG(count_baths) AS avg_count_full_bathroom
FROM
(
SELECT
    COUNT(*) as count_baths
FROM full_bathroom
NATURAL JOIN household
GROUP BY email
) AS subq
```

(31) Aggregate info for all commodes

```
SELECT
MAX(count_commodes) AS max_count_commodes,
MIN(count_commodes) AS min_count_commodes,
AVG(count_commodes) AS avg_count_commodes
FROM
(
SELECT
    SUM(number_of_commodes) AS count_commodes
FROM (
    SELECT email, bathroom_number, number_of_commodes
    FROM half_bathroom
    UNION
    SELECT email, bathroom_number, number_of_commodes
    FROM full_bathroom
) AS subq
NATURAL JOIN household
GROUP BY email
) AS main
```

(32) Aggregate info for all sinks

```
SELECT
MAX(count_sinks) AS max_count_sinks,
MIN(count_sinks) AS min_count_sinks,
AVG(count_sinks) AS avg_count_sinks
FROM
(
SELECT
    SUM(number_of_sinks) AS count_sinks
FROM (
    SELECT email, bathroom_number, number_of_sinks
    FROM half_bathroom
    UNION
    SELECT email, bathroom_number, number_of_sinks
    FROM full_bathroom
) AS subq
NATURAL JOIN household
GROUP BY email
) AS main
```

(33) Aggregate info for all bidets

```
SELECT
MAX(count_bidets) AS max_count_bidets,
MIN(count_bidets) AS min_count_bidets,
AVG(count_bidets) AS avg_count_bidets
FROM
(
SELECT
    SUM(number_of_bidets) AS count_bidets
FROM (
    SELECT email, bathroom_number, number_of_bidets
    FROM half_bathroom
    UNION
    SELECT email, bathroom_number, number_of_bidets
    FROM full_bathroom
) AS subq
NATURAL JOIN household
GROUP BY email
) AS main
```

(34) Aggregate info for all bathtubs

```
SELECT
MAX(count_bathtubs) AS max_count_bathtubs,
MIN(count_bathtubs) AS min_count_bathtubs,
```

```
AVG(count_bathtubs) AS avg_count_bathtubs
FROM
(
SELECT
    SUM(number_of_bathtubs) AS count_bathtubs
FROM full_bathroom
NATURAL JOIN household
GROUP BY email
) AS subq
```

(35) Aggregate info for all showers

```
SELECT
MAX(count_showers) AS max_count_showers,
MIN(count_showers) AS min_count_showers,
AVG(count_showers) AS avg_count_showers
FROM
(
SELECT
    SUM(number_of_showers) AS count_showers
FROM full_bathroom
NATURAL JOIN household
GROUP BY email
) AS subq
```

(36) Aggregate info for all tub/showers

```
SELECT
MAX(count_tub_showers) AS max_count_tub_showers,
MIN(count_tub_showers) AS min_count_tub_showers,
AVG(count_tub_showers) AS avg_count_tub_showers
FROM
(
SELECT
    COUNT(number_of_tub_showers) AS count_tub_showers
FROM full_bathroom
NATURAL JOIN household
GROUP BY email
) AS subq
```

(37) Select from city info and bathroom tables

```
SELECT
    state,
    SUM(number_of_bidets) AS total_bidets
FROM
(
    SELECT email, bathroom_number, number_of_bidets
```

```
    FROM half_bathroom
    UNION
    SELECT email, bathroom_number, number_of_bidets
    FROM full_bathroom
) AS subq
NATURAL JOIN household
NATURAL JOIN city_info
GROUP BY state
ORDER BY total_bidets DESC
LIMIT 1
```

(38) Select from city info and bathroom tables

```
SELECT
    postal_code,
    SUM(number_of_bidets) AS total_bidets
FROM (
    SELECT email, bathroom_number, number_of_bidets
    FROM half_bathroom
    UNION
    SELECT email, bathroom_number, number_of_bidets
    FROM full_bathroom
) AS subq
NATURAL JOIN household
NATURAL JOIN city_info
GROUP BY postal_code
ORDER BY total_bidets DESC
LIMIT 1
```

(39) Select from household and bathroom tables

```
SELECT
    COUNT(email) AS only_single_primary
FROM (
    SELECT
    email,
    is_primary_bathroom,
    COUNT(*) AS count_baths
    FROM (
        SELECT email, bathroom_number, false AS is_primary_bathroom
        FROM half_bathroom
        UNION
        SELECT email, bathroom_number, is_primary_bathroom
        FROM full_bathroom
    ) AS subq
) AS main
NATURAL JOIN household
WHERE count_baths = 1
AND is_primary_bathroom = true
GROUP BY email
```

Household Averages by Radius Report

Abstract Code

- User inputs (`$Center Postal Code`)
    - The (‘$Center Postal Code’) must be found in the postal code/city table, otherwise throw an error “ The postal code specified is invalid.” (40)
- User inputs (`$Radius`) from a drop-down menu of (0, 5, 10, 25, 50, 100, 250).
- Show postal code, search radius, and summary metrics from households that are located within (`$Radius`) miles from (`$Center Postal Code`) using the haversine within the DB.
    - Show aggregate of average bathroom count per household (41)
    - Show aggregate of average bedroom count per household (42)
    - Show aggregate of average occupant count per household (43)
    - Show aggregate of average ratio between commodes vs occupants (44)
    - Show aggregate of average appliances count per household (45)
    - Show aggregate of the most common heat source for any appliance with a heat source (46)

Queries

(40) Get Postal Code

```
    SELECT longitude, latitude FROM city_info
    WHERE postal_code = '$Center Postal Code'
```

Haversine

```
CREATE OR REPLACE FUNCTION haversine(lat1 float, lon1 float, lat2 float, lon2 float) R
ETURNS float AS $$
DECLARE
    radian_lat1 float := lat1 * pi() / 180;
    radian_lon1 float := lon1 * pi() / 180;
    radian_lat2 float := lat2 * pi() / 180;
    radian_lon2 float := lon2 * pi() / 180;

    dlon float := radian_lon2 - radian_lon1;
    dlat float := radian_lat2 - radian_lat1;
```

```
    a float := sin(dlat / 2) ^ 2 + cos(radian_lat1) * cos(radian_lat2) * sin(dlon / 2)
^ 2;
    c float := 2 * atan2(sqrt(a), sqrt(1 - a));
    d float := 3958.75 * c;

BEGIN
    RETURN d;
END;
```

(41) Average bathroom count per household

```
SELECT AVG(bathroom_count)
FROM
(
SELECT
    COUNT(*) as bedroom_count
FROM household
NATURAL JOIN city_info
LEFT JOIN (
    SELECT
        email,
        bathroom_number
    FROM
    (
    SELECT
        email,
        bathroom_number
    FROM half_bathroom
    UNION
    SELECT
        email,
        bathroom_number
    FROM full_bathroom
    ) AS subq
) AS subq_bathroom ON household.email = subq_bathroom.email
WHERE haversine(longitude, latitude, $Center Longitude, $Center Latitude) < $Radius
GROUP BY email
) AS bathroom_per_households
```

(42) Average bedrooms per household

```
SELECT
AVG(number_of_bedrooms)
FROM household
NATURAL JOIN city_info
WHERE haversine(longitude, latitude, $Center Longitude, $Center Latitude) < $Radius
```

(43) Average occuapnts per household

```
SELECT
AVG(number_of_occupants)
FROM household
NATURAL JOIN city_info
WHERE haversine(longitude, latitude, $Center Longitude, $Center Latitude) < $Radius
```

(44) Number of Commodes vs. Occupants

```
SELECT AVG(ratio_commodes_occupants)
FROM
(
SELECT
    email, number_of_occupants / SUM(number_of_commodes) as ratio_commodes_occupants
FROM household
NATURAL JOIN city_info
LEFT JOIN (
    SELECT
        email,
        bathroom_number,
        number_of_commodes
    FROM
    (
    SELECT
        email,
        bathroom_number,
        number_of_commodes
    FROM half_bathroom
    UNION
    SELECT
        email,
        bathroom_number,
        number_of_commodes
    FROM full_bathroom
    ) AS subq
) AS subq_bathroom ON household.email = subq_bathroom.email
WHERE haversine(longitude, latitude, $Center Longitude, $Center Latitude) < $Radius
GROUP BY email, number_of_occupants
) AS ratio_commodes_occupants
```

(45) Average appliances count per household

```
SELECT AVG(appliances_per_households)
FROM
(
SELECT
    COUNT(*) as bedroom_count
FROM household
NATURAL JOIN city_info
LEFT JOIN appliance
ON household.email = appliance.email
```

```
WHERE haversine(longitude, latitude, $Center Longitude, $Center Latitude) < $Radius
GROUP BY email
) AS appliances_per_households
```

(46) The most common heat source for any appliance with a heat source

```
SELECT
COUNT(heat_source)
(
    SELECT email, appliance_number, heat_source
    FROM dryer
    UNION
    SELECT email, appliance_number, cook_heat_source AS heat_source
    FROM cooktop
    UNION
    SELECT email, appliance_number, oven_heat_source AS heat_source
    FROM oven_heat_source
) AS heat_sources
WHERE haversine(longitude, latitude, $Center Longitude, $Center Latitude) < $Radius
NATURAL JOIN households
NATURAL JOIN city_info
GROUP BY heat_source
```