Table of Contents:

Task Decomposition with Abstract Code

Main Menu

Abstract Code

- Show "***Enter my household info***" and ***"View Reports/Query Data"*** buttons.
- If "***Enter my household info***" button is pushed, perform **Enter Household Info** task
- If "***View Reports/Query Data***" button is pushed, perform **View Report** task

Enter Household Info

Abstract Code

- User enters '$email', '$postal_code'
- User selects '$household_type' from the dropdown
- User enters '$square_footage'
- User ticks "No Heat" box or enters '$heating_setting'
- User ticks "No Cooling" box or enters '$cooling_setting'
- User checks PublicUtilities (not required) '$PublicUtility'
- If "***Next***" button is pushed,
    - **Validate User Inputs** as below:
        - If the '$email' input already exists in Household table (namely the below query returns count > 0):

```
SELECT COUNT(email) FROM `Household` WHERE email='$email';
```

            - Error message includes "Email already exists!"
        - If the '$postal_code' doesn't exist in Location table (namely the below query returns counts = 0):

```
SELECT COUNT(postal_code) FROM `Location` WHERE
postal_code='$postal_code';
```

            - Error message includes "Postal code is not correct!"
        - Elseif '$postal_code' exists in Location table (namely the above query returns counts = 1):
            - Display city and state etc information for confirmation

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

        - If data validation of h*ome_type, square_footage* (positive whole number) doesn't pass:

        - Error message includes “Please enter valid Home Type/Square footage”
    - If “No Heat” box is not ticked
        - If data validation of *Thermostat setting for heating* (as a whole degrees Fahrenheit) doesn't pass:
            - Error message includes “Please enter valid thermostat setting for heating”
    - If “No Cooling” box is not ticked
        - If data validation of *Thermostat setting for cooling* (as a whole degrees Fahrenheit) doesn't pass:
            - Error message includes “Please enter valid thermostat setting for cooling”
- If there is error message, **Pop Out Error Message Window and Go Back to Enter Household**
- If there is no error
    - Save Data To Household Table

```
INSERT INTO `Household` (email, square_footage, household_type, postal_code) VALUES ('$email', '$square_footage', '$household_type', '$postal_code');
```

    - If “No Heat” box is not ticked, Save Data to Household Table

```
INSERT INTO `Household` (email, heating_setting) VALUES ('$email', '$heating_setting');
```

    - If “No Cooling” box is not ticked, Save Data to Household Table

```
INSERT INTO `Household` (email, cooling_setting) VALUES ('$email', '$cooling_setting');
```

    - If some Public Utilities are selected, for every selected utility, Save Data to PublicUtility Table

```
INSERT INTO `PublicUtility` (email, public_utility) VALUES ('$email', '$public_utility');
```

    - **Go To Add Appliance**

Add Appliance

Abstract Code

- **Display Appliance Fields Based on User Selection**

- Display dropdown menu of appliance_type and user selects '$appliance_type', display dropdown menu for user selection of '$manufacturer_name', display model and BTU and user enters '$model' and '$BTU'
- if user selects *Air Handler*
    - Displays fields of RPM and user enters '$RPM'
    - Displays heating/cooling method of '$AirHandlerType'
        - If user selects *air conditioner,* displays *EER* and user enters '$EER'
        - If user selects *heater*, displays *energy source* and user enters '$energy_source'
        - If user selects *heat pup*, displays *SEER and HSPF* and user enters '$SEER' and '$HSPF'
- If user selects *Water Heater*
    - Displays fields of tank size, current temperature setting and energy source
    - User enters '$tank_size', '$current_temperature' and '$energy_source'

- If "***Add***" button is pushed,
    - **Run Data Validation of the Entered Fields**
        - If '$BTU' doesn't exist or not in whole number, add to error message
        - If '$manufacturer_name' *or* '$model' is Null, add to error message
        - If *Air Handler* is selected
            - if '$RPM' is Null or not in whole number, add to error message
            - If *air conditioner* is selected, '$EER' is Null or not in decimal number to the tenth decimal point, add to error message
            - if *heater* is selected and '$energy_source' is Null, add to error message
            - if *heat pup* is selected, '$SEER' or '$HSPF' *is Null* or not in decimal number to the tenth decimal point, add to error message
        - If *Water heater* is selected,
            - if '$tank_size' is Null *or* not in decimal number to the tenth decimal point, add to error message
            - if '$current_temperature' is not Null but not in whole number, add to error message
            - If '$energy_source' is Null, add to error message
        - If there is error message, **Pop Out Error Message Window** and **Go Back to Add Appliance**
    - **If there is no error, based on user inputs**

- **Create new Appliance to APPLIANCE database**

```
INSERT INTO `Appliance` (email, manufacturer_name, model, BTU)
VALUES ('$email', '$manufacturer_name', '$model', '$BTU');
```

  - Get last auto insert ID applianceID

```
SET @applianceID = SELECT LAST_INSERT_ID();
```

  - If '$ApplianceType' is Air Handler

```
INSERT INTO `AirHandler` (email, applianceID, RPM) VALUES
('$email', @applianceID, '$RPM');
```

  - If '$AirHanlderType' is Air Conditioner

```
INSERT INTO `AirConditioner` (email, applianceID, EER) VALUES
('$email', @applianceID, '$EER');
```

  - If '$AirHanlderType' is Heater

```
INSERT INTO `Heater` (email, applianceID, energy_source) VALUES
('$email', @applianceID, '$energy_source');
```

  - If '$AirHanlderType' is HeatPump

```
INSERT INTO `HeatPump` (email, applianceID, SEER, HSPF) VALUES
('$email', @applianceID, '$SEER', '$HSPF');
```

  - If '$ApplianceType' is Water Heater

```
INSERT INTO `WaterHeater` (email, applianceID, tank_size,
current_temperature, energy_source) VALUES ('$email',
@applianceID, '$tank_size', '$ current_temperature',
'$energy_source');
```

- **Then go to Appliance Listing**

<u>Appliance Listing</u>

Abstract Code

- **View Appliance**
  - Query the Appliance table and display a list of appliances with info of Type, Manufacturer and Model

```
SELECT applianceID, appliance_type, manufacturer_name, model
FROM `Appliance` WHERE email='$email';
```

- If **"*delete*"** button is pushed, the related record is to **Delete Appliance** from Appliance table

```
DELETE From `Appliance` where email='$email' AND applianceID='$applianceID';
```

- If **"*Add another appliance*"** button is pushed, go to **Add Appliance**

Jump to appliance form the page (as described above) and have the user input the appliance information.

- If **"*Next*"** button is pushed, go to **Add Power Generation**

Add Power Generation

Abstract Code

- Check if the user is Off-the-Grid
  - Only If '$email' is in PublicUtility table, then ***Skip*** button is present

```
SELECT COUNT(email) FROM `PublicUtility` WHERE email='$email';
```

  - If user presses ***Skip*** button, go to Power Generation Listing form
- If user presses ***Add button***
  - If *generation type* is not selected, add to error message
  - If *average monthly kilowatt hours* is Null or not in whole number, add to error message
  - If *battery storage capacity* is not Null but not in whole number, add to error message
- If there is error message, Pop Out Error Message Window and Go Back to Add Power Generation
- If there is no error, based on user inputs, Save Data to PowerGenerator table and go to Power Generation Listing

```
INSERT INTO `PublicUtility` (email, generator_type,
average_monthly_kilowatt_hours_generated,
battery_storage_capacity_kilowatt_hours) VALUES ('$email', '$generator_type',
'$average_monthly_kilowatt_hours_generated',
'$battery_storage_capacity_kilowatt_hours');
```

Power Generation Listing

Abstract Code

- **View Power Generation**
  - Query the PowerGenerator table and display a list of Power Generation with info of Type, Monthly Kwh and MoBattery kWh

```
SELECT generatorID, generator_type,
average_monthly_kilowatt_hours_generated,
battery_storage_capacity_kilowatt_hours FROM `PublicUtility`
where email='$email';
```

- If **"*delete*"** button is pushed, the related record is to **Delete Power Generation** from POWERGENERATOR table

```
DELETE From `PowerGenerator` where email='$email' AND generatorID='$generatorID';
```

- If **"*Add more power*"** button is pushed, go to Add Power Generation form
- If **"*Finish*"** button is pushed,
  - If there is no power added and the user is not in PublicUtility table, i.e., the result of following query returns 0, pop up Window to remind adding power, then go back to Add Power Generation form

```
SELECT COUNT(email)
FROM (
SELECT pg.email
FROM `PowerGenerator` pg LEFT Outer JOIN `PublicUtility` pu ON pg.email =
pu.email
WHERE pg.email='$email'
UNION
SELECT pu.email
FROM `PowerGenerator` pg RIGHT OUTER JOIN `PublicUtility` pu ON pg.email =
pu.email
WHERE pu.email='$email'
) AS derived_table_alias;
```

  - else:
    - Go to Wrapping Up form

Wrapping Up

Abstract Code

- Show "***Return to the main menu"*** button.
- If "***Return to the main menu***" button is pushed, perform **Main Menu** task

View reports

Abstract Code

- Show links for the following sections: ***Top 25 popular manufacturers, Manufacturer/model search, Heating/cooling method details, Water heater statistics by state, Off-the-grid household dashboard, Household averages by radius***
- If each of the link is pushed by the user, perform the corresponding task with the same name

  - *e.g.*, if "***Top 25 popular manufacturers***" is pushed, perform **Top 25 popular manufacturers**
- If "***Finish***" button is pushed, perform **Main Menu** task

Top 25 popular manufacturers

Abstract Code

- **List the top twenty-five manufacturers**
  - Query the APPLIANCE table and MANUFACTURER table (with APPLICANCE left joined by MANUFACTURER via common key ManufacturerName)
  - Display the list of top 25 manufacturers with the most appliances, with columns of manufacturer name, raw count presented (see code below)
  - At each row show a button ***drilldown report*** to enable the subtask **drilldown report**
  - If ***drilldown report*** pushed by the user, perform **drilldown report** for the specific manufacturer in that row

```
## this is the code for list top 25 manufacturer
SELECT Appliance.manufacturer_name, count(*) AS count_appliances
FROM Appliance INNER JOIN Manufacturer
ON Appliance.manufacturer_name =Manufacturer.manufacturer_name
GROUP BY manufacturer_name
ORDER BY manufacturer_name DESC
LIMIT 25;
```

- **Drilldown report**
  - Display manufacturer name at the top
  - Display a table with code as below

```
## assuming that the user select '$Manufacturer_Name'
SELECT * FROM
(
  SELECT 'WaterHeater' AS appliance_type, COUNT(*) AS count_appliances
  FROM Appliance AS A
  INNER JOIN WaterHeater AS WH ON A.email = WH.email AND A.applianceID
= WH.applianceID
  WHERE A.manufacturer_name = '$Manufacturer_Name'

  UNION ALL

  SELECT 'AirHandler' AS appliance_type, COUNT(*) AS count_appliances
  FROM Appliance AS A
```

```
  INNER JOIN AirHandler AS AH ON A.email = AH.email AND A.applianceID =
AH.applianceID
  WHERE A.manufacturer_name = '$Manufacturer_Name'

  UNION ALL

  SELECT 'AirConditioner' AS appliance_type, COUNT(*) AS
count_appliances
  FROM Appliance AS A
  INNER JOIN AirConditioner AS AC ON A.email = AC.email AND
A.applianceID = AC.applianceID
  WHERE A.manufacturer_name = '$Manufacturer_Name'

  UNION ALL

  SELECT 'Heater' AS appliance_type, COUNT(*) AS count_appliances
  FROM Appliance AS A
  INNER JOIN Heater AS HT ON A.email = HT.email AND A.applianceID =
HT.applianceID
  WHERE A.manufacturer_name = '$Manufacturer_Name'

  UNION ALL

  SELECT 'HeatPump' AS appliance_type, COUNT(*) AS count_appliances
  FROM Appliance AS A
  INNER JOIN HeatPump AS HP ON A.email = HP.email AND A.applianceID =
HP.applianceID
  WHERE A.manufacturer_name = '$Manufacturer_Name'
) AS subquery;
```

- If "***Finish***" button is pushed, perform **View Reports** task

<u>Manufacturer/model search</u>

Abstract Code

- User enters a string of *keyword* in the input field

```
## this is the code for Manufacturer/model search
## assume the user enter '$keyword'
SELECT model, manufacturer_name FROM Appliance
WHERE (model LIKE '%$keyword%' OR manufacturer_name LIKE '%$keyword%')
ORDER BY model ASC, manufacturer_name ASC;
```

- Highlight the any substring containing *keyword* in both columns (if any) of Model and <u>ManufacturerName</u>

- If "***Finish***" button is pushed, perform **View Reports** task

Heating/cooling method details

Abstract Code

- **Create temporary Tables**

```
## this is the code for Heating/cooling method details
SET TEMPORARY TABLE Heating_cooling AS
    (SELECT H.email, household_type, applianceID, BTU, RPM
    FROM Household AS H LEFT JOIN
        (SELECT A1.email, A1.applianceID, BTU, RPM
         FROM Appliance AS A1
         INNER JOIN AirHandler AS A2
         ON (A1.email = A2.email and
         A1.applianceID = A2.applianceID)) as A
    ON H.email = A.email);
CREATE TEMPORARY TABLE HC_AC AS
    (SELECT HC.email, HC.applianceID,
     household_type, RPM, BTU, EER
     FROM Heating_cooling as HC
     INNER JOIN AirConditioner as AC
     ON (HC.applianceID = AC.applianceID
     AND HC.email = AC.email));
CREATE TEMPORARY TABLE HC_HT AS
    (SELECT HC.email, HC.applianceID,
     household_type, RPM, BTU, energy_source
     FROM Heating_cooling as HC
     INNER JOIN Heater as HT
     ON (HC.applianceID = HT.applianceID
     AND HC.email = HT.email));
CREATE TEMPORARY TABLE HC_HP AS
    (SELECT HC.email, HC.applianceID,
     household_type, RPM, BTU, SEER, HSPF
     FROM Heating_cooling as HC
     INNER JOIN HeatPump as HP
     ON (HC.applianceID = HP.applianceID
     AND HC.email = HP.email));
```

- **Air Conditioner Summary**

```
SELECT H.household_type,
COALESCE(count_ac, 0) AS count_ac,
COALESCE(average_BTU_ac, 0) AS average_BTU_ac,
```

```
COALESCE(average_RPM_ac, 0) AS average_ RPM _ac,
COALESCE(average_EER_ac, 0) AS average_ EER _ac
FROM
      (SELECT DISTINCT(household_type) FROM Household) AS H
LEFT JOIN
      (SELECT household_type, count(*) AS count_ac,
      round(avg(BTU), 0) AS average_BTU_ac,
      round(avg(RPM),1) AS average_RPM_ac,
      round(avg(EER),1) AS average_EER_ac
      FROM HC_AC
      GROUP BY household_type) AC_summary
ON H.household_type = AC_summary.household_type
ORDER BY H.household_type ASC;
```

- **Heater Summary**

```
CREATE TEMPORARY TABLE common_energy_source AS
(SELECT household_type,
    energy_source AS most_common_energy_source
    FROM
        (SELECT household_type, energy_source,
        ROW_NUMBER() OVER (PARTITION BY household_type
        ORDER BY count_energy_source DESC) AS row_num
        FROM
            (SELECT household_type, energy_source,
            count(*) as count_energy_source
            FROM HC_HT
            GROUP BY household_type, energy_source)
            AS sub1)
        AS sub2
    WHERE row_num = 1);

SELECT H.household_type, COALESCE(count_ht, 0) AS count_ht,
COALESCE(average_BTU_ht, 0) AS average_BTU_ht,
COALESCE(average_RPM_ht, 0) AS average_RPM_ht,
most_common_energy_source FROM
(SELECT DISTINCT(household_type) FROM Household) AS H
LEFT JOIN
     (SELECT household_type, count(*) AS count_ht,
      round(avg(BTU), 0) AS average_BTU_ht,
      round(avg(RPM),1) AS average_RPM_ht
      FROM HC_HT GROUP BY household_type) HT_summary
ON H.household_type = HT_summary.household_type
LEFT JOIN common_energy_source
ON H.household_type = common_energy_source.household_type
ORDER BY H.household_type ASC;
```

- **Heat bump Summary**

```
SELECT H.household_type,
COALESCE(count_hp, 0) as count_hp,
COALESCE(average_BTU_hp, 0) as average_BTU_hp,
COALESCE(average_RPM_hp, 0) as average_RPM_hp,
COALESCE(average_SEER_hp, 0) as average_SEER_hp,
COALESCE(average_HSPF_hp, 0) as average_HSPF_hp
FROM
      (SELECT DISTINCT(household_type) FROM Household) H
LEFT JOIN
      (SELECT household_type, count(*) AS count_hp,
      round(avg(BTU), 0) AS average_BTU_hp,
      round(avg(RPM),1) AS average_RPM_hp,
      round(avg(SEER),1) AS average_SEER_hp,
      round(avg(HSPF),1) AS average_HSPF_hp
      FROM HC_HP
      GROUP BY household_type) HP_summary
ON H.household_type = HP_summary.household_type
ORDER BY H.household_type ASC;
```

- If "***Finish***" button is pushed, perform **View Reports** task

Water heater statistics by state

Abstract Code

- ***List water heater statistics by state***
    - Display query result:

```
SELECT tmp.state as state,
      ROUND(avg(tmp.tank_size),0) as avg_tank_size,
      ROUND(avg(tmp.BTU),0) as avg_BTU,
      ROUND(avg(tmp.current_temperature),1) as
      avg_current_temp,
      count(tmp.current_temperature) as count_provided_temp,
      count(*)-count(tmp.current_temperature) as
      count_not_provided_temp
      FROM
            (SELECT L.state, W.tank_size, W.BTU,
            W.current_temperature
            FROM Household as H
            LEFT JOIN Location as L
            ON H.postal_code = L.postal_code
```

```
            LEFT JOIN WaterHeater as W
            ON H.email= W.email
            LEFT JOIN Appliance as A
            ON W.email = A.email AND W.applianceID =
            A.applianceID) as tmp
      GROUP BY tmp.state
      ORDER BY tmp.state ASC;
```

- At each row show a button ***drilldown report*** to enable the subtask **drilldown report**
- If ***drilldown report*** pushed by the user, perform **drilldown report** for the specific state in that row, get the selected state value '$state'
- **Drilldown report**
    - Display state name at the top
    - Display query result:

```
SELECT tmp.energy_source,
       ROUND(min(tmp.tank_size),0) as min_tank_size,
       ROUND(avg(tmp.tank_size),0) as avg_tank_size,
       ROUND(max(tmp.tank_size),0) as max_tank_size,
       min(tmp.current_temperature) as min_temp_setting,
       ROUND(avg(tmp.current_temperature),1) as
      avg_current_temp,
       max(tmp.current_temperature) as max_temp_setting
       FROM
            (SELECT L.state, W.tank_size, W.energy_source,
            W.current_temperature
            FROM Household as H
            LEFT JOIN Location as L
            ON H.postal_code = L.postal_code
            LEFT JOIN WaterHeater as W
            ON H.email= W.email
            WHERE L.state= '$state') as tmp
      GROUP BY tmp.energy_source
      ORDER BY tmp.energy_source ASC;
```

- If "***Finish***" button is pushed, perform **View Reports** task

Off-the-grid household dashboard

Abstract Code

- Produce following tables via code：
  - **State with most off-grid**

```
SELECT tmp.state, count(*) as household_count
      FROM
      (SELECT L.state
      FROM Household as H
      LEFT JOIN Location as L
      ON H.postal_code = L.postal_code
      WHERE H.email NOT IN
           (SELECT email FROM PublicUtility)
      ) as tmp
GROUP BY tmp.state
ORDER BY count(*) DESC LIMIT 1;
```

  - **Average battery storage capacity per battery**

```
SELECT ROUND(avg(tmp.battery_storage_capacity_kilowatt_hours))
      as avg_battery_capacity
FROM
      (SELECT P.battery_storage_capacity_kilowatt_hours
      FROM Household as H
      LEFT JOIN PowerGenerator as P
      ON H.email = P.email
      WHERE H.email NOT IN
            (SELECT email FROM PublicUtility)
      ) as tmp;
```

  - **Percentage of power generation type**

```
SET @total_off_grid := (
  SELECT COUNT(*)
  FROM Household AS H
  LEFT JOIN PublicUtility AS PU ON H.email = PU.email
  WHERE PU.email IS NULL
);

SET @solar := (
  SELECT COUNT(*)
  FROM Household AS H
```

```
  LEFT JOIN PowerGenerator AS PG ON H.email = PG.email
  WHERE H.email NOT IN (SELECT email FROM PublicUtility)
  AND PG.generator_type = 'solar'
);

SET @wind := (
  SELECT COUNT(*)
  FROM Household AS H
  LEFT JOIN PowerGenerator AS PG ON H.email = PG.email
  WHERE H.email NOT IN (SELECT email FROM PublicUtility)
  AND PG.generator_type = 'wind-turbine'
);

SET @mixed := (
  SELECT COUNT(*)
  FROM Household AS H
  LEFT JOIN PowerGenerator AS PG ON H.email = PG.email
  WHERE H.email NOT IN (SELECT email FROM PublicUtility)
  AND PG.generator_type = 'mixed'
);

SET @solar_perc := ROUND((@solar / @total_off_grid) * 100, 1);
SET @wind_perc := ROUND((@wind / @total_off_grid) * 100, 1);
SET @mixed_perc := ROUND((@mixed / @total_off_grid) * 100, 1);

SELECT 'solar' AS generator_type, CONCAT(@solar_perc, '%') AS
percentage
UNION ALL
SELECT 'wind-turbine' AS generator_type, CONCAT(@wind_perc, '%') AS
percentage
UNION ALL
SELECT 'mixed' AS generator_type, CONCAT(@mixed_perc, '%') AS
percentage;
```

- **Percentage of household type**

```
SET @total_off_grid := (
  SELECT COUNT(*)
  FROM Household AS H
  WHERE H.email NOT IN (SELECT email FROM PublicUtility)
);

SET @house_count := (
  SELECT COUNT(CASE WHEN household_type = 'house' THEN 1 END)
  FROM Household AS H
```

```
  WHERE H.email NOT IN (SELECT email FROM PublicUtility)
);

SET @apartment_count := (
  SELECT COUNT(CASE WHEN household_type = 'apartment' THEN 1 END)
  FROM Household AS H
  WHERE H.email NOT IN (SELECT email FROM PublicUtility)
);

SET @townhome_count := (
  SELECT COUNT(CASE WHEN household_type = 'townhome' THEN 1 END)
  FROM Household AS H
  WHERE H.email NOT IN (SELECT email FROM PublicUtility)
);

SET @condominium_count := (
  SELECT COUNT(CASE WHEN household_type = 'condominium' THEN 1 END)
  FROM Household AS H
  WHERE H.email NOT IN (SELECT email FROM PublicUtility)
);

SET @modular_home_count := (
  SELECT COUNT(CASE WHEN household_type = 'modular home' THEN 1 END)
  FROM Household AS H
  WHERE H.email NOT IN (SELECT email FROM PublicUtility)
);

SET @tiny_house_count := (
  SELECT COUNT(CASE WHEN household_type = 'tiny house' THEN 1 END)
  FROM Household AS H
  WHERE H.email NOT IN (SELECT email FROM PublicUtility)
);

SET @total_types := @house_count + @apartment_count +
@townhome_count + @condominium_count + @modular_home_count +
@tiny_house_count;

SELECT
  'house' AS type,
  CONCAT(ROUND((@house_count / @total_off_grid) * 100, 1), '%') AS
count_percent
UNION ALL
SELECT
  'apartment' AS type,
```

```
  CONCAT(ROUND((@apartment_count / @total_off_grid) * 100, 1), '%')
AS count_percent
UNION ALL
SELECT
  'townhome' AS type,
  CONCAT(ROUND((@townhome_count / @total_off_grid) * 100, 1), '%')
AS count_percent
UNION ALL
SELECT
  'condominium' AS type,
  CONCAT(ROUND((@condominium_count / @total_off_grid) * 100, 1),
'%') AS count_percent
UNION ALL
SELECT
  'modular home' AS type,
  CONCAT(ROUND((@modular_home_count / @total_off_grid) * 100, 1),
'%') AS count_percent
UNION ALL
SELECT
  'tiny house' AS type,
  CONCAT(ROUND((@tiny_house_count / @total_off_grid) * 100, 1), '%')
AS count_percent;
```

- **Average water heater tank size**

```
SET @avg_off_grid := (
  SELECT ROUND(AVG(WH.tank_size), 1)
  FROM WaterHeater AS WH
  INNER JOIN Household AS H ON WH.email = H.email
  WHERE H.email NOT IN (SELECT email FROM PublicUtility)
);

SET @avg_on_grid := (
  SELECT ROUND(AVG(WH.tank_size), 1)
  FROM WaterHeater AS WH
  INNER JOIN Household AS H ON WH.email = H.email
  WHERE H.email IN (SELECT email FROM PublicUtility)
);

SELECT @avg_off_grid AS avg_off_grid, @avg_on_grid AS avg_on_grid;
```

- **Min, max, average BTU**

```
SET @min_waterheater := (
```

```
  SELECT ROUND(IFNULL(MIN(A.BTU), 0), 0)
  FROM Appliance AS A
  INNER JOIN Household AS H ON A.email = H.email
  INNER JOIN WaterHeater AS WH ON A.email = WH.email AND
A.applianceID = WH.applianceID
  WHERE H.email NOT IN (SELECT email FROM PublicUtility)
);

SET @avg_waterheater := (
  SELECT ROUND(IFNULL(AVG(A.BTU), 0), 0)
  FROM Appliance AS A
  INNER JOIN Household AS H ON A.email = H.email
  INNER JOIN WaterHeater AS WH ON A.email = WH.email AND
A.applianceID = WH.applianceID
  WHERE H.email NOT IN (SELECT email FROM PublicUtility)
);

SET @max_waterheater := (
  SELECT ROUND(IFNULL(MAX(A.BTU), 0), 0)
  FROM Appliance AS A
  INNER JOIN Household AS H ON A.email = H.email
  INNER JOIN WaterHeater AS WH ON A.email = WH.email AND
A.applianceID = WH.applianceID
  WHERE H.email NOT IN (SELECT email FROM PublicUtility)
);

SET @min_airconditioner := (
  SELECT ROUND(IFNULL(MIN(A.BTU), 0), 0)
  FROM Appliance AS A
  INNER JOIN AirConditioner AS AC ON A.email = AC.email AND
A.applianceID = AC.applianceID
  INNER JOIN Household AS H ON AC.email = H.email
  WHERE H.email NOT IN (SELECT email FROM PublicUtility)
);

SET @avg_airconditioner := (
  SELECT ROUND(IFNULL(AVG(A.BTU), 0), 0)
  FROM Appliance AS A
  INNER JOIN AirConditioner AS AC ON A.email = AC.email AND
A.applianceID = AC.applianceID
  INNER JOIN Household AS H ON AC.email = H.email
  WHERE H.email NOT IN (SELECT email FROM PublicUtility)
);
```

```
SET @max_airconditioner := (
  SELECT ROUND(IFNULL(MAX(A.BTU), 0), 0)
  FROM Appliance AS A
  INNER JOIN AirConditioner AS AC ON A.email = AC.email AND
A.applianceID = AC.applianceID
  INNER JOIN Household AS H ON AC.email = H.email
  WHERE H.email NOT IN (SELECT email FROM PublicUtility)
);

SET @min_heater := (
  SELECT ROUND(IFNULL(MIN(A.BTU), 0), 0)
  FROM Appliance AS A
  INNER JOIN Heater AS H ON A.email = H.email AND A.applianceID =
H.applianceID
  INNER JOIN Household AS HH ON H.email = HH.email
  WHERE HH.email NOT IN (SELECT email FROM PublicUtility)
);

SET @avg_heater := (
  SELECT ROUND(IFNULL(AVG(A.BTU), 0), 0)
  FROM Appliance AS A
  INNER JOIN Heater AS H ON A.email = H.email AND A.applianceID =
H.applianceID
  INNER JOIN Household AS HH ON H.email = HH.email
  WHERE HH.email NOT IN (SELECT email FROM PublicUtility)
);

SET @max_heater := (
  SELECT ROUND(IFNULL(MAX(A.BTU), 0), 0)
  FROM Appliance AS A
  INNER JOIN Heater AS H ON A.email = H.email AND A.applianceID =
H.applianceID
  INNER JOIN Household AS HH ON H.email = HH.email
  WHERE HH.email NOT IN (SELECT email FROM PublicUtility)
);

SET @min_heatpump := (
  SELECT ROUND(IFNULL(MIN(A.BTU), 0), 0)
  FROM Appliance AS A
  INNER JOIN HeatPump AS HP ON A.email = HP.email AND A.applianceID
= HP.applianceID
  INNER JOIN Household AS HHH ON HP.email = HHH.email
  WHERE HHH.email NOT IN (SELECT email FROM PublicUtility)
);
```

```
SET @avg_heatpump := (
  SELECT ROUND(IFNULL(AVG(A.BTU), 0), 0)
  FROM Appliance AS A
  INNER JOIN HeatPump AS HP ON A.email = HP.email AND A.applianceID
= HP.applianceID
  INNER JOIN Household AS HHH ON HP.email = HHH.email
  WHERE HHH.email NOT IN (SELECT email FROM PublicUtility)
);

SET @max_heatpump := (
  SELECT ROUND(IFNULL(MAX(A.BTU), 0), 0)
  FROM Appliance AS A
  INNER JOIN HeatPump AS HP ON A.email = HP.email AND A.applianceID
= HP.applianceID
  INNER JOIN Household AS HHH ON HP.email = HHH.email
  WHERE HHH.email NOT IN (SELECT email FROM PublicUtility)
);

SELECT
  'WaterHeater' AS type,
  @min_waterheater AS min_BTU,
  @avg_waterheater AS avg_BTU,
  @max_waterheater AS max_BTU
UNION ALL
SELECT
  'AirConditioner' AS type,
  @min_airconditioner AS min_BTU,
  @avg_airconditioner AS avg_BTU,
  @max_airconditioner AS max_BTU
UNION ALL
SELECT
  'Heater' AS type,
  @min_heater AS min_BTU,
  @avg_heater AS avg_BTU,
  @max_heater AS max_BTU
UNION ALL
SELECT
  'HeatPump' AS type,
  @min_heatpump AS min_BTU,
  @avg_heatpump AS avg_BTU,
  @max_heatpump AS max_BTU;
```

Household averages by radius

Abstract Code

- User enters string of '$postal_code' and '$radius' in the input field

- If postal code entered is invalid, display error message "invalid postal code" (namely the below query returns counts = 0):

```
SELECT count(postal_code)
FROM Location
WHERE postal_code='$postal_code';
```

- Retrieve input postal_code's latitude and longitude value:

```
SET @target_latitude:=(
      SELECT latitude FROM Location WHERE postal_code= '$postal_code');
SET @target_longitude:=(
      SELECT longitude FROM Location WHERE postal_code= '$postal_code');
```

- Write a function called calc_distance

```
DELIMITER $$
CREATE FUNCTION calc_distance(
      lat1 DECIMAL,
      lon1 DECIMAL,
      lat2 DECIMAL,
      lon2 DECIMAL
)
RETURNS REAL
DETERMINISTIC
BEGIN
      DECLARE a REAL;
      DECLARE c REAL;
      DECLARE d REAL;
      SET a = POWER(SIN((lat2-lat1)/2),2)+COS(lat1)*COS(lat2)*
POWER(SIN((lon2-lon1)/2),2);
      SET c = 2*ATAN2(SQRT(a),SQRT(1-a));
      SET d = 3958.75*c;
      RETURN d;
END$$
DELIMITER;
```

- Create intermediate tables for household within the selected radius

```
CREATE TABLE Location_distance AS
```

```
      SELECT L.postal_code, calc_distance(L.latitude,L.longitude,
             @target_latitude, @target_longitude) as distance
      FROM Location as L;

CREATE TABLE In_radius_household AS
      SELECT H.email, H.household_type, H.square_footage,
       H.heating_setting, H.cooling_setting
       FROM Household as H
       LEFT JOIN Location_distance as L
       ON H.postal_code = L.postal_code
       WHERE L.distance <= '$radius';

CREATE TABLE In_radius_household_with_power AS
      SELECT H.email, P.generator_type,
             P.average_monthly_kilowatt_hours_generated,
             P.battery_storage_capacity_kilowatt_hours
       FROM Household as H
       LEFT JOIN Location_distance as L
       ON H.postal_code = L.postal_code
       INNER JOIN PowerGenerator as P
       ON H.email = P.email
       WHERE L.distance <= '$radius';

CREATE TABLE In_radius_most_common_power AS
      SELECT generator_type
      FROM In_radius_household_with_power
      GROUP BY generator_type
      ORDER BY count(*) DESC LIMIT 1;

CREATE TABLE In_radius_power_with_battery AS
      SELECT email
      FROM In_radius_household_with_power
      WHERE battery_storage_capacity_kilowatt_hours IS NOT NULL;

CREATE TABLE In_radius_household_with_utility AS
      SELECT H.email, P.public_utility
      FROM Household as H
      LEFT JOIN Location_distance as L
      ON H.postal_code = L.postal_code
```

```
        LEFT JOIN PublicUtility as P
        ON H.email = P.email
        WHERE L.distance <= '$radius';
```

- Return query result:

```
WITH H AS (
        SELECT '$postal_code' as postal_code,
        '$radius' as search_radius,
        COALESCE(count(DISTINCT IRH.email),0) AS count_houshold,
        SUM(IRH.household_type= 'house') AS count_house,
        SUM (IRH.household_type= 'apartment') AS count_apartment,
        SUM (IRH.household_type= 'townhome') AS count_townhome,
        SUM (IRH.household_type= 'condominium') AS count_condominium,
        SUM (IRH.household_type= 'modular home') AS count_modular_home,
        SUM (IRH.household_type= 'tiny house') AS count_tiny_house,
        ROUND(AVG(IRH.square_footage),0) AS avg_footage,
        ROUND (AVG(IRH.heating_setting),1) AS avg_heat_temp,
        ROUND (AVG(IRH.cooling_setting),1) AS avg_cool_temp
        FROM In_radius_household AS IRH),
HU AS (
       SELECT '$postal_code' AS postal_code,
       GROUP_CONCAT(DISTINCT (IRU.public_utility) SEPARATOR ',') AS
              public_utilities,
       count(IRU.email)- count(IRU.public_utility) AS num_off_grid
       FROM In_radius_household_with_utility AS IRU),
HP AS (
       SELECT '$postal_code' AS postal_code,
      COALESCE (count(DISTINCT IRP.email),0) AS count_with_power,
      ROUND (AVG(IRP.average_monthly_kilowatt_hours_generated),0) AS
       avg_monthly_power
       FROM In_radius_household_with_power AS IRP),
CP AS (
       SELECT '$postal_code' AS postal_code,
       GROUP_CONCAT(DISTINCT (IRCP. generator_type) SEPARATOR '') AS
       most_common_generator_method
       FROM In_radius_most_common_power AS IRCP),
PB AS (
       SELECT '$postal_code' AS postal_code,
       COALESCE (count(DISTINCT IRPB.email),0) AS count_with_battery
       FROM In_radius_power_with_battery AS as IRPB)
SELECT H.postal_code, H.search_radius, H.count_houshold, H.count_house,
H.count_apartment, H.count_townhome, H.count_condominium,
H.count_modular_home, H.count_tiny_house, H.avg_footage, H.avg_heat_temp,
```

```
H.avg_cool_temp, HU.public_utilities, HU.num_off_grid,
HP.count_with_power, CP.most_common_generator_method,
HP.avg_monthly_power, PB.count_with_battery
FROM H
LEFT JOIN HU
ON H.postal_code = HU.postal_code
LEFT JOIN HP
ON H.postal_code = HP.postal_code
LEFT JOIN CP
ON H.postal_code = CP.postal_code
LEFT JOIN PB
ON H.postal_code = PB.postal_code;
```

- If "***Finish***" button is pushed, perform **View Reports** task