Alternakraft Phase 2 Report

CS 6400 Database Systems and Concepts

Georgia Institute of Technology

Summer 2023

Team #08:

Yadhap Dahal, Thomas Gracie, Sheldon Greiling, Garrison Winter

Table of Contents

Table of Contents

Task Decomposition and Abstract Code

Task #1 Enter Household Info:

Abstract Code

(Called from main menu, when enter household link is clicked)

1. Display household info from
2. When ***Next button*** is clicked, validate user inputs
    - email = must pass html5 email validation check
    - postalCode = must be of type text and maximum length 5
    - household_type = must be either house, apartment, townhome, condominium, modular home, or tiny home
    - square_footage = must be whole number
    - heating = must be whole number unless 'no heat' checkbox is checked
    - cooling = must be whole number unless 'no cooling' checkbox is checked
    - PublicUtilities = must be zero or more of these: Electric, Gas, Steam, Liquid Fuel

3. If validation fail:
    - Display appropriate error message

4. If validation check passed, attempt data insertion into Household table
    - Display error if email already exists in Database
    - Display error if postal code in not in Database
    - Display error if any error occurs

```
INSERT INTO Household(email, postalCode, household_type, square_footage, heating, cooling)
VALUES('$email',  '$postalCode', '$household_type', '$square_footage', '$heating', '$cooling');
```

5. If household info inserted, attempt data insertion into PublicUtilityLinkage table

- Display error if any error occurs
- Display 'add appliances form' upon successful insertion

```
INSERT INTO PublicUtilityLinkage(householdID, public_utility_id)
VALUES('$householdID', '$public_utility_id');
```

Task #2 Add Appliances:

Abstract Code

(Called upon successful insertion of household details into the database or when 'add another appliance' is clicked from list appliance component)

1. Get list of all manufactures and their ID from database

```
SELECT manufacturerID, manufacturer_name FROM Manufacturer;
```

2. Display 'Add appliance form' with single select component to select appliance type
3. When the value for appliance type selected, display following form fields:
    - Manufacturer
    - Model name
    - BTU Rating
4. If application type is 'water heater', display form fields to collect below information
    - Tank size
    - Temperature Setting
    - Energy source
5. If application type is 'air handler', display form fields to collect below information
    - Fan rotation per minute (RPM)
    - Heating and cooling method
6. For heating and cooling method:
    - If 'air conditioner' selected, display input field for Energy Efficiency ratio (EER)
    - If 'heater' selected, display input field for Energy Source
    - If 'heat pump' selected, display input field for SEER and HSPF
7. When the ***add button*** is clicked, validate user inputs and attempt data insertion in appropriate table. If validation fails display an appropriate error message.
    - manufacturer must be a string
    - model_name is optional, when entered it must be string
    - btu_rating must be a whole number

```
INSERT INTO Appliance(householdID, applianceID, applianceType, manufacturerID, model_name,
btu_rating)
VALUES('$householdID', '$applianceID', '$applianceType', '$manufacturerID', '$model_name',
'$btu_rating');
```

if appliance type selected is ‘water heater’, insert data into WaterHeater table

- tank_size must be decimal value
- temperature can be null or whole number
- energy_source must be either electric, gas, fuel oil or heat pump

```
INSERT INTO WaterHeater(householdID, applianceID, energy_source, tank_size, temp_setting)
VALUES('$householdID', '$applianceID', '$energy_source', 'tank_size', 'temp_setting');
```

if appliance type selected is ‘air handler’

```
INSERT INTO AirHandler(HouseholdID, applianceID, rpm)
VALUES('$HouseholdID', '$applianceID', '$rpm');
```

If heating /cooling method is ‘air conditioner’:

- eer must be decimal number

```
INSERT INTO AirConditioner(householdID, applianceID, eer)
VALUES('$householdID', '$applianceID', '$eer');
```

If heating /cooling method is ‘heater’:

- energy_source must be either electric, gas or thermosolar

```
INSERT INTO Heater(householdID, applianceID, `source`)
VALUES('$householdID', '$applianceID', '$source');
```

heating /cooling method is ‘heat pump’

- SEER and HSPF must be decimal values

```
INSERT INTO HeatPump(householdID, applianceID, seer, hspf)
VALUES('$householdID', '$applianceID', '$seer', '$hspf')
```

8. Display appropriate errors if any error occurs during data insertion, if no errors take user to ‘list appliances component’ (task #3)

Task #3 List Appliances:

Abstract Code

(Called when a new appliance is successfully added)

1. Display all appliance that were recently added in tabular format,

```
SELECT  a.applianceID, a.applianceType, m.manufacturer_name, a.model_name
FROM Appliance AS a
INNER JOIN Manufacturer AS m ON a.manufacturerID = m.manufacturerID
WHERE a.householdID = '$householdID';
```

2. provide option to delete appliance, add another appliance and go to next step
3. When ***delete button*** is clicked, take user to task #4
4. When 'add another appliance' link is clicked, take user to task #2
5. When ***next button*** is clicked display form to add power generation

Task #4 Delete Appliances:

Abstract Code

(Called when delete button is clicked from 'List Appliance' component)

1. When ***delete button*** is clicked, attempt to delete appliance from the database

```
DELETE FROM Appliance
WHERE applianceID = '$applianceID' AND householdID  = '$householdID';
```

- If deletion successful – Display success message and refresh appliance table
- If deletion is unsuccessful – Display failure message

Task #5 Add Power Gen:

Abstract Code

(Called by clicking next button from list appliance component)

1. Display power generation form.
    - If Select statement is null disable skip button on add power generation form

```
SELECT householdID
FROM Household
WHERE '$householdID' NOT IN (SELECT householdID FROM PublicUtilityLinkage);
```

2. Provide option to add power generation information

```
-- Get Power Generator Types
SELECT generatorTypeID, generator_type
FROM PowerGeneratorType;
```

3. Provide option to skip this task if household is on-the-grid
4. When ***skip button*** is clicked
    - Display submission complete message and a link to main menu
5. When ***add button*** is clicked, validate user inputs
    - type must be solar or wind-turbine
    - monthly_kwh must be whole number
    - storage_kwh must be optional, must be a whole number if provided
6. If validation fails, display appropriate error message.
7. If validation successful, attempt to add power generation information to the database
    - If insertion successful – take user to 'Power Generation List' component (task#6)
    - If insertion fails – Display appropriate message

```
-- Insert new powergenerator into table
INSERT INTO PowerGenerator (householdID, powerGeneratorID, generatorTypeID,
monthly_kilowatt, battery_storage)
VALUES ('$householdID', '$powerGeneratorID', '$generatorTypeID', '$monthly_kilowatt',
'$battery_storage');
```

Task #6 List Power Gen:

Abstract Code

(Called upon successful insertion of new power generation method)

1. Display all recently added power generation in tabular format
2. provide option to delete, add more power and go to next step
3. When ***delete button*** is clicked, take user to Task #7
4. When 'add more power' link is clicked, take user to Task #5
5. When **next button** is clicked display submission complete message and a link to main menu

```
-- List househould powergeneration

SELECT PG.powerGeneratorID, PGT.generator_type, PG.monthly_kilowatt, PG.battery_storage
FROM PowerGenerator PG
JOIN PowerGeneratorType PGT ON PG.generatorTypeID = PGT.generatorTypeID
```

Task #7 Delete Power Gen:

Abstract Code

(Called when delete button is clicked from 'List power generation' component)

1. When ***delete button*** is clicked, attempt to delete power generation information from the database

- If deletion successful – Display success message and refresh power generation table
- If deletion is unsuccessful – Display appropriate failure message

```
-- Delete instance of househould powergeneration
DELETE
FROM PowerGenerator
WHERE householdID = '$householdID' and powerGeneratorID = '$powerGeneratorID';
```

Task #8 View Top25 Pop Manufg:

Abstract Code

(Called when '25 top manufacturers' link is clicked from all reports page)

1. Calculate top 25 manufacturers based on number of appliances

```
SELECT m.manufacturer_name AS Manufacturer, COUNT(app.applianceID) AS TotalAppliances
FROM Manufacturer m
JOIN Appliance app ON m.manufacturerID = app.manufacturerID
GROUP BY m.manufacturer_name
ORDER BY TotalAppliances DESC
LIMIT 25;
```

2. Display result in tabular format, make the name of manufacturer clickable.
3. When the manufacturer name is clicked:
    - Calculate total number of appliances, categorized by appliance type for the given manufacturer
    - Display the result in tabular format

```
SELECT app.applianceType  AS ApplianceType, COUNT(app.applianceID) AS TotalAppliances
FROM Manufacturer m
JOIN Appliance app ON m.manufacturerID = app.manufacturerID AND m.manufacturer_name =
'$Manufacturer_Name'
GROUP BY app.applianceType;
```

4. Provide options to go to main menu and all reports page

Task #9 Search/View Manufg/Model:

Abstract Code

(Called when 'Manufacturer / Modal search' link is clicked from all reports page)

1. Provide input field for entering search string
2. When search string is entered:
    - Perform database search to find any manufacturer or model that match the provided string
    - Present result in tabular format. Highlight matched model/manufacturer
    - If no matching result found, inform the user

```
SELECT m.manufacturer_name AS Manufacturer, app.model_name AS Model
FROM Manufacturer m
JOIN Appliance app ON m.manufacturerID = app.manufacturerID
WHERE m.manufacturer_name LIKE '%$SearchString%' OR app.model_name LIKE
'%$SearchString%';
```

Task #10 View Heating/Cooling Details:

Abstract Code

(Called when 'view heating/cooling method detail' link is clicked from all reports page)

Display data in 3 different tables:

1. Air conditioners
   - Calculate the total number of air conditioners for each kind of household
   - Calculate air conditioners' average BTU, RPM, EER for each kind of household
   - Display above result in tabular format, order data by household type

```
SELECT DISTINCT hh1.household_type AS HouseHoldType,
    CASE
            WHEN avg_ac.TotalAirConditioners IS NULL THEN 0
             ELSE avg_ac.TotalAirConditioners
    END AS TotalAirConditioners,
    CASE
             WHEN avg_ac.Average_BTU IS NULL THEN 0
             ELSE avg_ac.Average_BTU
    END AS Average_BTU,
    CASE
             WHEN avg_ac.Average_RPM IS NULL THEN 0
             ELSE avg_ac.Average_RPM
    END AS Average_RPM,
    CASE
            WHEN avg_ac.Average_EER IS NULL THEN 0
            ELSE avg_ac.Average_EER
    END AS Average_EER
   FROM
   Household hh1
left join (
             SELECT
             hh.household_type AS HouseHoldType, COUNT(app.applianceID) AS TotalAirConditioners
            ,round(AVG(app.btu_rating),0) AS Average_BTU, round(AVG(ah.rpm),1) AS Average_RPM
            ,round(AVG(ac.eer),1) AS Average_EER
            FROM Household hh
            JOIN Appliance app ON hh.householdID = app.householdID
            JOIN AirHandler ah ON ah.householdID = app.householdID AND ah.applianceID = app.applianceID
            JOIN AirConditioner ac ON ac.householdID = app.householdID AND ac.applianceID =
            app.applianceID
            GROUP BY hh.household_type
            ) as avg_ac on hh1.household_type = avg_ac.HouseHoldType;
```

2. Heaters
    - Calculate the total number of heaters for each kind of household
    - Compute Heaters' average BTU and RPM values for each kind of household
    - Determine the most widely used energy source for operating heaters for each type of household
    - Display above result in tabular format, order data by household type

```
SELECT DISTINCT hh1.household_type AS HouseHoldType,
    CASE
            WHEN avg_h.Total_Heaters IS NULL THEN 0
            ELSE avg_h.Total_Heaters
    END AS Total_Heaters,
    CASE
            WHEN avg_h.Average_BTU IS NULL THEN 0
            ELSE avg_h.Average_BTU
    END AS Average_BTU,
    CASE
            WHEN avg_h.Average_RPM IS NULL THEN 0
            ELSE avg_h.Average_RPM
    END AS Average_RPM,
    CASE
            WHEN avg_h.Most_Used_Energy_Source IS NULL THEN 'N/A'
            ELSE avg_h.Most_Used_Energy_Source
    END AS Most_Used_Energy_Source
FROM
   Household hh1
LEFT JOIN (
              SELECT
              hh.household_type AS HouseHoldType, COUNT(*) AS Total_Heaters
              ,round(AVG(app.btu_rating),0) AS Average_BTU, round(AVG(ah.rpm),1) AS Average_RPM
              ,MAX(stc.source) AS Most_Used_Energy_Source
              FROM
              Household hh
              JOIN Appliance app ON hh.householdID = app.householdID
              JOIN AirHandler ah ON ah.householdID = app.householdID AND ah.applianceID = app.applianceID
              JOIN Heater hON h.householdID = ah.householdID AND h.applianceID = ah.applianceID
              JOIN
              (
              SELECT householdID, applianceID, `source`, COUNT(*) AS source_type_count
              FROM Heater
              GROUP BY householdID, applianceID, `source`
               ) AS stc ON h.householdID = stc.householdID AND h.applianceID = stc.applianceID
               GROUP BY hh.household_type
      ) as avg_h on hh1.household_type = avg_h.HouseHoldType;
```

3. Heat pumps
    - Calculate total number of heat pump for each kind of household
    - Calculate heat pump's average BTU, RPM, SEER and HSPF for each kind of household
    - Display above result in tabular format, order data by household type

```
SELECT DISTINCT hh1.household_type AS HouseHoldType,
   CASE
         WHEN avg_hp.TotalHeatPumps IS NULL THEN 0
         ELSE avg_hp.TotalHeatPumps
   END AS TotalHeatPumps,
   CASE
         WHEN avg_hp.Average_BTU IS NULL THEN 0
         ELSE avg_hp.Average_BTU
   END AS Average_BTU,
   CASE
         WHEN avg_hp.Average_RPM IS NULL THEN 0
         ELSE avg_hp.Average_RPM
   END AS Average_RPM,
   CASE
         WHEN avg_hp.Average_SEER IS NULL THEN 0
         ELSE avg_hp.Average_SEER
   END AS Average_SEER,
   CASE
         WHEN avg_hp.Average_HPSF IS NULL THEN 0
         ELSE avg_hp.Average_HPSF
   END AS Average_HPSF
FROM
  Household hh1
LEFT JOIN (
        SELECT hh.household_type AS HouseHoldType ,COUNT(*) AS TotalHeatPumps
          ,round(AVG(app.btu_rating),0) AS Average_BTU, round(AVG(ah.rpm),1) AS Average_RPM
          ,round(AVG(hp.seer),1) AS Average_SEER, round(AVG(hp.hspf),1) AS Average_HPSF
        FROM Household hh
        JOIN Appliance app ON hh.householdID = app.applianceID
        JOIN AirHandler ah ON ah.householdID = app.householdID AND ah.applianceID = app.applianceID
        JOIN HeatPump hp ON hp.householdID = app.householdID AND hp.applianceID = app.applianceID
        GROUP BY hh.household_type
    ) as avg_hp on hh1.household_type = avg_hp.HouseHoldType;
```

4. Provide options to go to main menu and all reports page

Task #11 View Water Heater Stats by State:

Abstract Code

(Called when 'Water Heater Statistics' link is clicked from all reports page)

1. Water Heater statistics for each state
    - Calculate the average water heater tank
    - Calculate the average water heater BTU
    - Calculate the average water heater temperature setting
    - Calculate the count of water heaters for which the temperature settings have been provided
    - Calculate the count of water heaters for which the temperature settings have not been provided

    Display above information in tabular format, make the state abbreviation clickable

```
SELECT
pc.state AS State, AVG(wh.tank_size) AS Average_Tank_Size, AVG(app.btu_rating) AS Average_BTU
,AVG(wh.temp_setting) AS Average_Temperature_Setting
,SUM(CASE WHEN wh.temp_setting IS NOT NULL THEN 1 ELSE 0 END) AS
Count_With_Temperature_Settings
,SUM(CASE WHEN wh.temp_setting IS NULL THEN 1 ELSE 0 END) AS
Count_Without_Temperature_Settings
FROM Household hh
JOIN PostalCode pc ON hh.postalCode = pc.postalCode
JOIN Appliance app ON hh.householdID = app.householdID
LEFT JOIN WaterHeater wh ON app.householdID = wh.householdID AND app.applianceID =
wh.applianceID
GROUP BY pc.state;
```

2. Provide options to go to main menu and all reports page

Task #12 View Water Heater Stats by State Drilldown:

Abstract Code

(Called when state abbreviation is clicked from water heater statistics table (Task #11))

1. When the state abbreviation is clicked
    - Calculate the minimum water heater tank size for the selected state
    - Calculate the maximum water heater tank size for the selected state
    - Calculate the average water heater tank size for the selected state
    - Calculate the minimum temperature settings for the selected state
    - Calculate the average temperature settings for the selected state
    - Calculate the maximum temperature settings for the selected state
    - Display the above information in tabular format

```
SELECT
MIN(wh.tank_size) AS MinTankSize, MAX(wh.tank_size) AS MaxTankSize
,ROUND(AVG(wh.tank_size),2)   AS AvgTankSize, MIN(wh.temp_setting) AS MinTempSetting
,MAX(wh.temp_setting) AS MaxTempSetting
,ROUND(AVG(wh.temp_setting),2) AS AvgTempSetting
FROM WaterHeater wh
JOIN Appliance app ON wh.householdID = app.householdID AND wh.applianceID = app.applianceID
JOIN Household hh ON app.householdID = hh.householdID
JOIN PostalCode pc ON hh.postalCode = pc.postalCode AND pc.state = '$State';
```

2. Provide options to go to main menu and all reports page

Task #13 View Off-the-Grid Details:

This task shows 6 different tables, using the subtasks described below. There is an option to go to the Main Menu and All Reports page.

Subtask #13.1 Top Off-the-grid by state:

Abstract Code

1. Determine the state with the most off-the-grid households and the count
2. Calculate the average battery storage capacity per battery, round it to a whole number
3. Display above data in single table

```
SELECT
pc.state AS State, COUNT(*) AS OffGridHouseholds
,ROUND(AVG(pg.battery_storage)) AS AverageBatteryStorage
FROM Household h
JOIN PostalCode pc ON h.postalCode = pc.postalCode
JOIN PowerGenerator pg ON h.householdID = pg.householdID
WHERE h.householdID NOT IN (SELECT householdID FROM PublicUtilityLinkage)
GROUP BY pc.state
ORDER BY OffGridHouseholds DESC
LIMIT 1;
```

Subtask #13.2 Off-the-grid AVG battery storage capacity:

Abstract Code

1. Calculate the proportion of power generation from each source for all off-the-grid households as a percentage.
2. Display the data in tabular format

```
SELECT
pg.generator_type AS GeneratorType, COUNT(*) AS OffGridHouseholds
,ROUND(SUM(pg.battery_storage)/
   (SELECT SUM(pg2.battery_storage)
     FROM PowerGenerator pg2) * 100,2) AS ProportionOfBatteryStorage
FROM Household h
JOIN PowerGenerator pg  ON h.householdID = pg.householdID
WHERE  h.heating IS NULL AND h.cooling IS NULL AND pg.battery_storage IS NOT NULL
GROUP BY pg.generator_type
ORDER BY OffGridHouseholds DESC;
```

Subtask #13.3 Off-the-grid Power Generator type breakdown:

Abstract Code

1. Calculate total number of off-the-grid households grouped by household type
2. If a particular type of house does not have any off-the-grid household display 0
3. Display this data in tabular format

```
SELECT
CASE
        WHEN count_solar = 0 THEN '0.0%'
        ELSE CONCAT(ROUND(100 * count_solar / (count_solar + count_windturbine + count_mixed), 1),
        '%')
END AS Solar,
CASE
        WHEN count_windturbine = 0 THEN '0.0%' ELSE CONCAT(ROUND(100 * count_windturbine /
        (count_solar + count_windturbine + count_mixed), 1), '%')
        END AS Windturbine,
CASE
        WHEN count_mixed = 0 THEN '0.0%' ELSE CONCAT(ROUND(100 * count_mixed / (count_solar +
        count_windturbine + count_mixed), 1), '%')
        END AS Mixed
FROM
(
SELECT COUNT(DISTINCT pg.householdID) AS count_solar
FROM PowerGenerator pg, Household h
WHERE pg.householdId = h.householdId
AND pg.householdId NOT IN (SELECT householdID FROM PublicUtilityLinkage)
AND pg.householdId IN (SELECT householdID FROM PowerGenerator WHERE generatorTypeID = 1)
AND pg.householdId NOT IN (SELECT householdID FROM PowerGenerator WHERE generatorTypeID != 1
) s,
(
SELECT COUNT(DISTINCT pg.householdID) AS count_windturbine
FROM PowerGenerator pg, Household h
WHERE pg.householdId = h.householdId
AND pg.householdId NOT IN (SELECT householdID FROM PublicUtilityLinkage)
AND pg.householdId IN (SELECT householdID FROM PowerGenerator WHERE generatorTypeID = 2)
AND pg.householdId NOT IN (SELECT householdID FROM PowerGenerator WHERE generatorTypeID != 2)
) wt,
(
SELECT COUNT(DISTINCT pg.householdID) AS count_mixed
FROM PowerGenerator pg, Household h
WHERE pg.householdId = h.householdId
AND pg.householdId NOT IN (SELECT householdID FROM PublicUtilityLinkage)
AND pg.householdId IN (SELECT householdID FROM PowerGenerator WHERE generatorTypeID = 1)
AND pg.householdId IN (SELECT householdID FROM PowerGenerator WHERE generatorTypeID = 2)
) m
```

Table of Contents

Subtask #13.4 Off-the-grid Household type breakdown:

Abstract Code

1. Calculate the average tank capacity of water heaters used in off-the-grid households
2. Calculate the average tank capacity of water heaters used in on-the-grid households
3. Display above data in tabular format

```
SELECT household_type, COUNT(
CASE
        WHEN householdId NOT IN (SELECT householdID FROM PublicUtilityLinkage)
        THEN 1 ELSE NULL
        END) AS num_off_the_grid
FROM Household h
GROUP BY h.household_type;
```

Subtask #13.5 Average water heater tank size by off-the-grid flag:

Abstract Code

1. Calculate the average water heater tank size for all water heaters in households which are off-the-grid.
2. Calculate the average water heater tank size for all water heaters in households which are on-the-grid.

```
SELECT on_the_grid_avg_waterheater_size, off_the_grid_avg_waterheater_size
FROM
(
SELECT AVG(wh.tank_size) AS on_the_grid_avg_waterheater_size
FROM WaterHeater wh, Household h
WHERE wh.householdID = h.householdID
AND wh.householdID IN (SELECT householdID FROM PublicUtilityLinkage)
) offgrid,
(
SELECT AVG(wh.tank_size) AS off_the_grid_avg_waterheater_size
FROM WaterHeater wh, Household h
WHERE wh.householdID = h.householdID
AND wh.householdID NOT IN (SELECT householdID FROM PublicUtilityLinkage)
) ongrid
```

Subtask #13.6 Off-the-Grid BTU Stats by Appliance Type:

Abstract Code

1. Calculate min, max and average BTU values for all off-the-grid households' appliances grouped by appliance type
2. If no data for specific appliance type, display zero
3. Display above data in tabular format

```
SELECT
app.applianceType AS ApplianceType, MIN(COALESCE(app.btu_rating, 0))  AS MinBTU
,MAX(COALESCE(app.btu_rating, 0))  AS MaxBTU
,ROUND(COALESCE(AVG(app.btu_rating), 0), 2) AS AvgBTU
FROM Appliance app
JOIN Household hh ON app.householdID = hh.householdID
WHERE hh.heating IS NULL AND hh.cooling IS NULL
GROUP BY app.applianceType;
```

Task #14 Household Averages by Radius:

Subtask #14.1 Get postal codes in radius:

1. User inputs postal code and maximum distance.
    a. The postal code must be valid.
    b. If a distance of 0 is entered, all postal codes are returned.

```
SELECT T4.SecondPostal
FROM (
   SELECT
    T3.MainPostal, T3.SecondPostal, 3958.75 * T3.c as Distance, T3.DistanceVariable
   FROM (
      SELECT
      T2.MainPostal, T2.SecondPostal, 2 * ATAN2(SQRT(T2.a), SQRT(1 - T2.a)) AS c          ,T2.DistanceVariable
      FROM (
         SELECT
         T1.MainPostal, T1.SecondPostal, SIN(RADIANS(T1.DeltaLat) / 2) *
          SIN(RADIANS(T1.DeltaLat) / 2) +  COS(RADIANS(T1.lat1)) * COS(RADIANS(T1.lat2)) *
          SIN(RADIANS(T1.DeltaLon) / 2) *      SIN(RADIANS(T1.DeltaLon) / 2) AS a,T1.DistanceVariable
         FROM
            (SELECT
                p1.PostalCode AS MainPostal, p2.PostalCode AS SecondPostal, p1.latitude AS lat1
               ,p2.latitude AS lat2, p2.latitude-p1.latitude AS DeltaLat, p1.longitude AS lon1
               ,p2.longitude AS lon2, p2.longitude-p1.longitude AS DeltaLon
                ,'$Distance' AS DistanceVariable
            FROM PostalCode p1
            CROSS JOIN PostalCode p2 ON p2.PostalCode != p1.PostalCode
             AND p1.PostalCode = '$PostalCode'

            UNION
           SELECT
                p1.PostalCode AS MainPostal, p1.PostalCode AS SecondPostal, p1.latitude AS lat1
               ,p1.latitude AS lat2, p1.latitude-p1.latitude AS DeltaLat, p1.longitude AS lon1
               ,p1.longitude AS lon2, p1.longitude-p1.longitude AS DeltaLon, 0 AS DistanceVariable
            FROM PostalCode p1 WHERE p1.PostalCode = '$PostalCode'
            )AS T1
         )AS T2
      )AS T3
)AS T4
WHERE Distance <= T4.DistanceVariable
```

Subtask #14.2 Household details by postal codes:

Abstract Code

(Called when 'household Average by Radius' link is clicked from all reports page)

1. Provide users with 2 input fields one for postal code and one for radius and search button
2. When ***search button*** is clicked, validate the postal code and search radius
3. If user input is invalid, display appropriate message
4. Use the haversine formula to calculate the straight-line distance between the postal code's coordinates and the coordinates of other postal codes within the search radius.
5. Retrieve the household data for all postal codes within the calculated distance.

```
WITH HAV
AS
(SELECT T4.SecondPostal
FROM (
  SELECT T3.MainPostal, T3.SecondPostal, 3958.75 * T3.c as Distance, T3.DistanceVariable
  FROM (
    SELECT T2.MainPostal, T2.SecondPostal, 2 * ATAN2(SQRT(T2.a), SQRT(1 - T2.a)) AS c,
T2.DistanceVariable
    FROM (
      SELECT T1.MainPostal, T1.SecondPostal, SIN(RADIANS(T1.DeltaLat) / 2) * SIN(RADIANS(T1.DeltaLat) /
         2) +  COS(RADIANS(T1.lat1)) * COS(RADIANS(T1.lat2)) * SIN(RADIANS(T1.DeltaLon) / 2) *
         SIN(RADIANS(T1.DeltaLon) / 2) AS a, T1.DistanceVariable
      FROM
        (SELECT
            p1.PostalCode AS MainPostal, p2.PostalCode AS SecondPostal, p1.latitude AS lat1 ,p2.latitude
        AS lat2, p2.latitude-p1.latitude AS DeltaLat, p1.longitude AS lon1, p2.longitude AS lon2,
        p2.longitude-p1.longitude AS DeltaLon, '$Distance' AS DistanceVariable
        FROM PostalCode p1
        CROSS JOIN PostalCode p2 ON p2.PostalCode != p1.PostalCode AND p1.PostalCode = '$PostalCode'
UNION
        SELECT
           p1.PostalCode AS MainPostal, p1.PostalCode AS SecondPostal, p1.latitude AS lat1
          ,p1.latitude AS lat2, p1.latitude-p1.latitude AS DeltaLat, p1.longitude AS lon1, p1.longitude AS
           lon2
          ,p1.longitude-p1.longitude AS DeltaLon,0 AS DistanceVariable
        FROM PostalCode p1 WHERE p1.PostalCode = '$PostalCode'
        )AS T1
      )AS T2
    )AS T3
)AS T4
WHERE Distance <= T4.DistanceVariable
)
```

6. Calculate the following statistics for the retrieved household data:
    - Count the total number of households.

```
SELECT COUNT(*)
FROM Household h, HAV hav
WHERE h.postalCode = hav.SecondPostal;
```

- Count the number of households for each household type, displaying 0 if there are none for a particular type.

```
SELECT household_type,
COUNT(CASE
        WHEN householdId IN (SELECT householdID FROM Household h, HAV hav WHERE h.postalCode =
        hav.SecondPostal)
        THEN 1
        ELSE NULL
END) AS households_of_type
FROM Household h
GROUP BY h.household_type;
```

- Calculate the average square footage of households as a whole number, rounded.

```
SELECT ROUND(AVG(square_footage), 0) AS avg_square_footage
FROM Household h, HAV hav
WHERE h.postalCode = hav.SecondPostal;
```

- Calculate the average heating temperature as a decimal number rounded to tenths.

```
SELECT ROUND(AVG(heating), 1) AS avg_heating_temp
FROM Household h, HAV hav
WHERE h.postalCode = hav.SecondPostal;
```

- Calculate the average cooling temperature as a decimal number rounded to tenths.

```
SELECT ROUND(AVG(cooling), 1) AS avg_cooling_temp
FROM Household h, HAV hav
WHERE h.postalCode = hav.SecondPostal;
```

- Determine which public utilities are used, displayed in a single cell and separated by commas.

```
SELECT GROUP_CONCAT(DISTINCT pu.public_utility) AS public_utilities_used
FROM Household h, HAV hav, PublicUtilityLinkage pul, PublicUtilities pu
WHERE h.postalCode = hav.SecondPostal
AND pul.householdID = h.householdID
AND pul.public_utility_id = pu.public_utility_id;
```

- Count the number of "off-the-grid" homes.

```
SELECT COUNT(CASE
        WHEN householdId NOT IN (SELECT householdID FROM PublicUtilityLinkage)
        THEN h.householdID
        ELSE NULL
END) AS num_off_grid_homes
FROM Household h, HAV hav
WHERE h.postalCode = hav.SecondPostal;
```

- Count the number of homes with power generation.

```
SELECT COUNT(DISTINCT h.householdID) AS num_homes_with_power_generation
FROM Household h, HAV hav, PowerGenerator pg
WHERE h.postalCode = hav.SecondPostal
AND h.householdID = pg.householdID;
```

- Determine the most common generation method for households with power generation.

```
SELECT pgt.generator_type
FROM Household h, HAV hav, PowerGenerator pg, PowerGeneratorType pgt
WHERE h.postalCode = hav.SecondPostal
AND h.householdID = pg.householdID
AND pg.generatorTypeID = pgt.generatorTypeID
GROUP BY pg.generatorTypeID
ORDER BY COUNT(h.householdID) DESC
LIMIT 1;
```

- Calculate the average monthly power generation per household as a whole number, rounded.

```
SELECT ROUND(AVG(pg_per_household), 0) AS avg_monthly_power_gen_per_household
FROM
(
SELECT SUM(pg.monthly_kilowatt) AS pg_per_household
FROM Household h, HAV hav, PowerGenerator pg
WHERE h.postalCode = hav.SecondPostal
AND h.householdID = pg.householdID
GROUP BY h.householdID
) f;
```

- Count the number of households with battery storage.

```
SELECT COUNT(DISTINCT pg.householdID) AS num_households_with_battery_storage
FROM Household h, HAV hav, PowerGenerator pg
WHERE h.postalCode = hav.SecondPostal
AND h.householdID = pg.householdID
AND pg.battery_storage IS NOT NULL;
```

- Display above data in tabular format

7. Provide option to go back to main menu and all reports page