Table of Contents:

Abstract Code with SQL Queries

Main menu

- Two links are shown:
  - When the ***Enter my household info*** link is clicked, go to the **Enter Household Info** page.
  - When the ***View reports/query data*** link is clicked, go to the **Reports** page.

Enter Household Info

Abstract Code

- User enters *email* ('$Email'), *postal code* ('$PostalCode'), *home type* ('$HomeType'), *square footage* ('$SquareFeet'), *thermostat setting for heating* ('$ThermostatHeating') or *no heat* ('$NoHeat'), *thermostat setting for cooling* ('$ThermostatCooling') or *no cooling* ('$NoCooling'), and *public utilities* ('$PublicUtilities') input fields.
- If data validation is successful for *email, postal code, home type, square footage, thermostat setting for heating* or *no heat, thermostat setting for cooling* or *no cooling,* and *public utilities* input fields, then:
  - When ***Next*** button is clicked:
    - If '$Email' is found in Household:
      - Highlight the *email* field with an error message.

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

    - If '$PostalCode' does not exist in Location:
      - Highlight the *postal code* field with an error message.

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

    - If '$ThermostatHeating' is blank but '$NoHeat' is unchecked:
      - Highlight both *thermostat setting for heating* and *no heat* with error styling.
    - If '$ThermostatCooling' is blank but '$NoCooling' is unchecked:
      - Highlight both *thermostat setting for heating* and *no heat* with error styling.
    - Else:
      - Store *email* ('$Email'), *postal code* ('$PostalCode'), *home type* ('$HomeType'), *square footage* ('$SquareFeet'), *thermostat setting for heating* ('$ThermostatHeating'), *thermostat setting for cooling* ('$ThermostatCooling') input fields (household information) as a new row in Household.

```
INSERT INTO Household (email, square_footage,
thermostat_cooling, thermostat_heating, type, postal_code)
VALUES ('$Email', '$SquareFootage', '$ThermostatCooling',
```

Table of Contents

```
'$ThermostatHeating', '$HomeType', '$PostalCode');
```

- For each *public utility (‘$PublicUtility’)* selected:
  - Store *email* ('$Email') and *public utilities* (‘$PublicUtilities’) input fields as new rows in PublicUtility:

```
INSERT INTO PublicUtility (household_email, name)
VALUES ('$Email', '$PublicUtility');
```

- Go to **Add Appliance** form.

- Else *email, zip code, home type, square footage, thermostat setting for heating, no heating, thermostat setting for cooling,* and/or *no cooling* input fields are invalid, display **Enter household info** form, with the error highlighted on the input field that failed data validation and the ***Next*** button disabled until errors are cleared.

Add Appliance

Abstract Code

- User picks the *appliance type* ('$ApplianceType’) option from a dropdown input field with options “Air handler” and “Water heater”.
- Populate the '$ApplianceManufacturer’ field with allowed values from the database by reading appliance manufacturers from ApplianceManufacturer.

```
SELECT name
FROM ApplianceManufacturer;
```

- Show the *BTU Rate* (‘$ApplianceBTU’), *appliance manufacturer* ('$ApplianceManufacturer’), and *model name* (‘$ApplianceModelName’) input fields.
- Show fields specific to each appliance type:
  - If '$ApplianceType’ == “Air handler”, show the following input fields:
    - All three AirHandler types are represented in Heating/Cooling methods as (“Air conditioner”, “Heater”, “Heat pump”) and all are allowed to be selected in any combination.
    - *Heating/cooling method* (‘$ApplianceAirHandlerHeatingCoolingMethod’)
      - If $ApplianceAirHandlerHeatingCoolingMethod == “Air conditioner”, show the *energy efficiency ratio* ('$ApplianceAirConditionerEER’) input field.
      - If $ApplianceAirHandlerHeatingCoolingMethod == “Heater”, show the *energy source* ('$ApplianceHeaterEnergySource’) input field with “Electric”, “gas”, and “thermosolar” as options.
      - If $ApplianceAirHandlerHeatingCoolingMethod == “Heat pump”, show the *seasonal energy efficiency rating* ('$ApplianceHeatPumpSEER’) and *heating seasonal performance factor* ('$ApplianceHeatPumpHSPF’) input fields.
    - *Fan rotations per minute* (‘$ApplianceAirHandlerFanRPM’)
    - *Energy efficiency ratio* (‘$ApplianceAirHandlerEnergyEfficiencyRatio’)
  - Else, if '$ApplianceType’ == “Water heater”, show the following input fields:
    - *Energy source* (‘$ApplianceWaterHeaterEnergySource’) with “Electric”, “gas”, “fuel oil”, or “heat pump” as options
    - *Tank size* (‘$ApplianceWaterHeaterTankSize’)

Table of Contents

        - *BTU rating* (‘$ApplianceWaterHeaterBTU’)
        - *Temperature setting* (‘$ApplianceWaterHeaterTemperature’)
- If data validation is successful for the combination of (*appliance manufacturer, BTU, model name) AND either (Heating/cooling method, Fan rotations per minute, Energy efficiency ratio OR Energy source, Tank size, BTU rating, Temperature setting)* input fields, then:
    - When ***Add*** button is clicked, calculate the new *appliance number* (‘$ApplianceNumber’) by reading the locally-stored number of appliances already existing for the current household.

```
SELECT COUNT(*) FROM
(SELECT appliance_number
FROM AirHandler
WHERE household_email='$Email'
UNION ALL
SELECT appliance_number
FROM WaterHeater
WHERE household_email='$Email')
all_appliances;
```

    - Then, store all input fields (appliance information) as a new row in Appliance based on the following conditions:
        - If '$ApplianceType’ == “Air handler”:

```
INSERT INTO AirHandler (appliance_number,
household_email, model_name, btu_rating,
fan_rotation, manufacturer_name)
VALUES ('$ApplianceNumber', '$Email',
'$ApplianceModelName', '$ApplianceBTU',
'$ApplianceAirHandlerFanRPM',
'$ApplianceManufacturer');
```

AND

```
INSERT INTO AirConditioner (appliance_number,
household_email, energy_efficiency_ratio)
VALUES ('$ApplianceNumber', '$Email',
'$ApplianceAirHandlerEnergyEfficiencyRatio');
```

AND

```
INSERT INTO Heater (appliance_number,
household_email, energy_source)
VALUES ('$ApplianceNumber', '$Email',
'$ApplianceHeaterEnergySource');
```

AND

```
INSERT INTO HeatPump (appliance_number,
household_email, seasonal_energy_efficiency_ratio,
heating_seasonal_performance_factor)
VALUES ('$ApplianceNumber', '$Email',
'$ApplianceHeatPumpSEER', '$ApplianceHeatPumpHSPF');
```

Table of Contents

        - Else, if '$ApplianceType' == “Water heater”:

```
INSERT INTO WaterHeater (appliance_number,
household_email, model_name, btu_rating, tank_size,
energy_source, temperature_rating,
manufacturer_name)
VALUES ('$ApplianceNumber', '$Email',
'$ApplianceModelName', '$ApplianceBTU',
'$ApplianceWaterHeaterTankSize',
'$ApplianceWaterHeaterEnergySource',
'$ApplianceWaterHeaterTemperature',
'$ApplianceManufacturer');
```

        - When successful, go to the **Appliance Listing** page.
- Else the combination of *(appliance manufacturer, BTU, model name)* AND either *(Heating/cooling method, Fan rotations per minute, Energy efficiency ratio* OR *Energy source, Tank size, BTU rating, Temperature setting)* input fields are invalid, display **Add Appliance** form, with error highlighted on the input field that failed data validation.

Appliance Listing

Abstract Code

- We make a Read call to generate a list of *appliances* (‘$Appliance’) with each *appliance*’s *number* (‘$ApplianceNumber’), *appliance type* (‘$ApplianceType’), *appliance manufacturer* ('$ApplianceManufacturer’), *appliance model* (‘$ApplianceModelName’), and a ***Delete*** button.
    - If the ***Delete*** button is clicked:
        - If *appliance type* (‘$ApplianceType’) == “Air handler”, we delete the *appliance* from several tables:

```
DELETE
FROM AirHandler
WHERE appliance_number='$ApplianceNumber' AND
household_email='$Email';
```

AND

```
DELETE
FROM AirConditioner
WHERE appliance_number='$ApplianceNumber' AND
household_email='$Email';
```

AND

```
DELETE
FROM Heater
WHERE appliance_number='$ApplianceNumber' AND
household_email='$Email';
```

AND

```
DELETE
```

Table of Contents

```
FROM HeatPump
WHERE appliance_number='$ApplianceNumber' AND
household_email='$Email';
```

    - Else, if *appliance type* (‘$ApplianceType’) == “Water heater”, we delete the *appliance* from WaterHeater:

```
DELETE
FROM WaterHeater
WHERE appliance_number='$ApplianceNumber' AND
household_email='$Email';
```

- The ***Add another appliance*** button displays the **Add Appliance** form to input a new *appliance*.
- When ***Next*** button is clicked:
    - Go to the **Add Power Generation** page.

Add Power Generation

Abstract Code

- Using the *email* ('$Email') input value, read PublicUtility for the household to see if there are any previously-stored values. Store the results locally to use later, if needed.

```
SELECT Count(*)
FROM PublicUtility
WHERE household_email='$Email';
```

    - If *public utilities* (‘$PublicUtilities’) has stored values, ***Skip*** button is displayed.
        - When ***Skip*** button is clicked:
            - Go to the **Thank You** page.
- User picks the *power generation type* ('$PowerGeneratorType’) option from a dropdown input field.
- If data validation is successful for the combination of (*power generation* (*type* (‘$PowerGenerationType’), *monthly kwh* (‘$MonthlyKwh’), *storage kwh* (‘$StorageKwh’)), then:
    - When ***Add*** button is clicked:
        - Using the locally-stored value of counts, calculate the *power generator number* (‘$PowerGeneratorNumber) by adding one.
        - Store all input fields (power generation information) as a new row in PowerGenerator.

```
INSERT INTO PowerGenerator (power_generator_number, email,
avg_monthly_kwh,  battery_storage_capacity,  type)
VALUES ('$PowerGeneratorNumber', '$Email', '$MonthlyKwh',
'$StorageKwh', '$PowerGeneratorType');
```

            - Go to the **Power Generation Listing** page.
- Else the combination of (*power generation type, monthly kwh, storage kwh)* input fields are invalid, display **Add Power Generation** form, with error highlighted on the input field that failed data validation.

Table of Contents

Power Generation Listing

Abstract Code

- We make a Read call to generate a list of *power generations* (‘$PowerGenerators’) with each *power generation number* (‘$PowerGeneratorNumber’), *power generation type* (‘$PowerGeneratorType’), *monthly kwh* ('$MonthlykWh’), *storage kwh* (‘$StoragekWh’), and a ***Delete*** button.
    - Each ***Delete*** button makes a Write call to `PowerGenerator` to remove a *power generator* from the *household*.

```
DELETE
FROM PowerGenerator
WHERE power_generator_number='$PowerGeneratorNumber' AND
household_email='$Email';
```

    - If the last Power Generator is deleted, and the household is off-grid, the ***Next*** button should not be enabled.
- The ***Add more power*** button displays the **Add Power Generation** form to input a new *power generator*.
- When ***Finish*** button is clicked:
    - Go to the **Thank You** page.

Thank You

Abstract Code

- One link is shown:
    - When the ***Return to the main menu*** link is clicked, go to the **Main menu** page.

Reports

Abstract Code

- A list of report names are linked to each report.
- When the ***Top 25 popular manufacturers*** link is clicked, go to the **Top 25 popular manufacturers** page.
- When the ***Manufacturer/model search*** link is clicked, go to the **Manufacturer/model search** page.
- When the ***Heating/cooling method details*** link is clicked, go to the **Heating/cooling method details** page.
- When the ***Water heater statistics by state*** link is clicked, go to the **Water heater statistics by state** page.
- When the ***Off-the-grid household dashboard*** link is clicked, go to the **Off-the-grid household dashboard** page.
- When the ***Household averages by radius*** link is clicked, go to the **Household averages by radius** page.
- When user clicks the ***Back*** button, show the **Main menu** page.

Table of Contents

Top 25 popular manufacturers

Abstract Code

- Read Appliance, Read on ApplianceManufacturer.
- The query returns a list of $ApplianceManufacturer and appliance count by $ApplianceManufacturer sorted descending by appliance count.

```
SELECT manufacturer_name, COUNT(*) as count_appliance
FROM
(SELECT manufacturer_name
FROM WaterHeater
UNION ALL
SELECT manufacturer_name
FROM AirHandler) as Manufacturer
GROUP BY manufacturer_name
ORDER BY count_appliance DESC
LIMIT 25;
```

  - Clicking on a link on each manufacturer opens a drill down report
    - The drill down report shows the $ApplianceType and the count of appliances by each $ApplianceType for the particular $ApplianceManufacturer.
      - If the count of an appliance type for the selected manufacturer is 0, show the appliance type anyway.

```
SELECT AM.name,
COUNT(W.appliance_number) AS count_WaterHeater,
COUNT(AH.appliance_number) AS count_AirHandler
FROM ApplianceManufacturer AS AM
LEFT OUTER JOIN WaterHeater as W ON AM.name=
W.manufacturer_name
LEFT OUTER JOIN AirHandler as AH ON AM.name=
AH.manufacturer_name
WHERE AM.name='$ApplianceManufacturer'
GROUP BY AM.name
ORDER BY AM.name;
```

- If the read does not return results, a message indicating “No Records Found” will be displayed.
- When user clicks the ***Back*** button, show the **Reports** page.

Manufacturer/model search

Abstract Code

- User enters a search *string* ('$SearchString’).
- Use the *search string* ('$SearchString’) to search Appliance for a case-insensitive partial-string match on the *appliance manufacturer* ('$ApplianceManufacturer’) column or the *appliance_model* (‘$ApplianceModelName’) column.

Table of Contents

    - Case-insensitive partial means a wildcard before and after the *search string* ('$SearchString').
- If the query returns results, there will be a list of query results ordered by *appliance manufacturer* ('$ApplianceManufacturer') ascending and *appliance model* ('$ApplianceModelName') ascending.

```
SELECT manufacturer_name, model_name
FROM
(SELECT manufacturer_name, model_name
FROM WaterHeater
UNION
SELECT manufacturer_name, model_name
FROM AirHandler) as Appliance
WHERE manufacturer_name LIKE '%$SearchString%' or model_name
LIKE '%$SearchString%';
```

    - The *appliance manufacturer* ('$ApplianceManufacturer') cell and/or the *appliance model* ('$ApplianceModelName') that partially or completely matches the *search string* ('$SearchString') is highlighted with a light green background.
- If the read does not return results, a message indicating "No Records Found" will be displayed.
- When user clicks the ***Back*** button, show the **Reports** page.

Heating/cooling method details

Abstract Code

- Read Household and Appliance.
- If the query returns results, show a table grouped and ordered by *home type* ('$HomeType') that displays various statistics for '$ApplianceType' == "Air handler":
    - For air handlers where $ApplianceAirHandlerHeatingCoolingMethod == "Air conditioner":
        - The count of air conditioners from AirConditioner, average air conditioner *BTU Rate* ('$ApplianceBTU') from AirHandler (as a whole number, rounded), average *Fan rotations per minute* ('$ApplianceAirHandlerFanRPM') from AirHandler (as a decimal number, rounded to tenths), and the average *energy efficiency ratio* ('$ApplianceAirConditionerEER') from AirConditioner (as a decimal number, rounded to tenths)

```
SELECT
H.type as home_type,
COUNT(AC.appliance_number) AS count_air_conditioners,
ROUND(AVG(AH.btu_rating), 0) as avg_BTU_rate,
FORMAT(ROUND(AVG(AH.fan_rotation), 1), 1) as avg_fan_rotation,
FORMAT(ROUND(AVG(AC.energy_efficiency_ratio), 1), 1) as
avg_energy_efficiency_ratio
FROM Household H
LEFT JOIN AirConditioner AC ON H.email=AC.household_email
LEFT JOIN AirHandler AH ON H.email=AH.household_email and
AH.appliance_number=AC.appliance_number
```

Table of Contents

```
GROUP BY home_type
ORDER BY home_type;
```

- For air handlers where $ApplianceAirHandlerHeatingCoolingMethod == “Heater”:
  - The count of heaters from Heater, average heater *BTU Rate* (‘$ApplianceBTU’) (as a whole number, rounded) from AirHandler, average *Fan rotations per minute* (‘$ApplianceAirHandlerFanRPM’) (as a decimal number, rounded to tenths) from AirHandler, and the Top 1 *energy source* ('$ApplianceHeaterEnergySource’) by count from Heater.

```
SELECT
H.type as home_type,
COUNT(HE.appliance_number) AS count_heaters,
ROUND(AVG(AH.btu_rating), 0) as avg_BTU_rate,
FORMAT(ROUND(AVG(AH.fan_rotation), 1), 1) as avg_fan_rotation,
(SELECT top_energy_source FROM
      (SELECT HE2.energy_source AS top_energy_source, COUNT(*)
      FROM Heater HE2
      INNER JOIN Household H2 ON H2.email=HE2.household_email
      WHERE H2.`type`=H.`type`
      GROUP BY HE2.energy_source
      ORDER BY 2 DESC
      LIMIT 1) as top_source) as top_energy_source
FROM Household H
LEFT JOIN Heater HE ON H.email=HE.household_email
LEFT JOIN AirHandler AH ON AH.household_email=H.email AND
AH.appliance_number=HE.appliance_number
GROUP BY home_type
ORDER BY home_type;
```

- For air handlers where $ApplianceAirHandlerHeatingCoolingMethod == “Heat pump”:
  - The count of heat pumps from HeatPump, average heat pump *BTU Rate* (‘$ApplianceBTU’) (as a whole number, rounded) from AirHandler, average *Fan rotations per minute* (‘$ApplianceAirHandlerFanRPM’) (as a decimal number, rounded to tenths) from AirHandler, the average *seasonal energy efficiency rating* ('$ApplianceHeatPumpSEER’) (as a decimal number, rounded to tenths) from HeatPump, and the average *heating seasonal performance factor* ('$ApplianceHeatPumpHSPF’) (as a decimal number, rounded to tenths) from HeatPump.

```
SELECT
H.type as home_type,
COUNT(HP.appliance_number) as count_heat_pump,
ROUND(AVG(AH.btu_rating), 0) as avg_BTU_rate,
FORMAT(ROUND(AVG(AH.fan_rotation), 1), 1) as avg_fan_rotation,
FORMAT(ROUND(AVG(HP.seasonal_energy_efficiency_ratio), 1), 1) as
avg_heat_pump_SEER,
FORMAT(ROUND(AVG(HP.heating_seasonal_performance_factor), 1), 1) as
avg_heat_pump_HSPF
```

Table of Contents

```
FROM Household H
LEFT JOIN HeatPump HP ON H.email=HP.household_email
LEFT JOIN AirHandler AH ON H.email=AH.household_email AND
AH.appliance_number=HP.appliance_number
GROUP BY home_type
ORDER BY home_type;
```

- If the read does not return results, all household types should be displayed even if the type does not have the appliance.
    - When user clicks the ***Back*** button, show the **Reports** page.

Water heater statistics by state

Abstract Code

- Read Household, Location, and Appliance.

```
SELECT l.state,
ROUND(AVG(w.tank_size), 0) as avg_tank_size,
ROUND(AVG(w.btu_rating), 0) as avg_btu_rating,
FORMAT(ROUND(AVG(w.temperature_setting), 1), 1) as
avg_temp_setting,
COUNT(w.temperature_setting) as count_temp_setting,
(COUNT(w.appliance_number) - COUNT(w.temperature_setting))
as count_no_temp_setting FROM Location l
LEFT OUTER JOIN Household h on h.postal_code=l.postal_code
LEFT OUTER JOIN WaterHeater w on h.email=w.household_email
GROUP BY l.state
ORDER BY l.state ASC;
```

- The query returns a list of appliances with $ApplianceType' == "Water heater" and average values of *Tank size* ('$ApplianceWaterHeaterTankSize') (as a whole number, rounded) from WaterHeater, *BTU rating* ('$ApplianceWaterHeaterBTU') (as a whole number, rounded) from WaterHeater, and *Temperature setting* ('$ApplianceWaterHeaterTemperature') (as a decimal number, rounded to tenths) from WaterHeater grouped by state. Display these results sorted by state abbreviation ascending.
    - If there are no water heaters and/or households for a state, the state should be displayed on this report with blank values for all statistical columns.
    - If the user clicks on a link in a row, another read on Appliance is initiated and:
        - The query returns a drilldown of $ApplianceType' == "Water heater" for the selected state, listing the *Energy source* ('$ApplianceWaterHeaterEnergySource') from WaterHeater; the minimum, average, and maximum (all as whole numbers, rounded) *Tank size* ('$ApplianceWaterHeaterTankSize') from WaterHeater; and the minimum, average (as a decimal number, rounded to tenths), and maximum *Temperature setting* ('$ApplianceWaterHeaterTemperature') from WaterHeater. Energy sources should be ordered in ascending order.
        - If the selected state has no water heaters utilizing that energy source:
            - All energy sources should be displayed (with blank values for any

statistical columns).

```
SELECT
est.source_type,
ROUND(MIN(w.tank_size), 0),
ROUND(AVG(w.tank_size), 0),
ROUND(MAX(w.tank_size), 0),
ROUND(MIN(w.temperature_setting), 1),
ROUND(AVG(w.temperature_setting), 1),
ROUND(MAX(w.temperature_setting), 1)
FROM WaterHeater w
INNER JOIN Household h ON h.email=w.household_email
INNER JOIN Location l ON l.postal_code=h.postal_code
RIGHT JOIN (SELECT distinct(energy_source) source_type,
household_email FROM WaterHeater) est
ON est.household_email=h.email
AND l.state='$State'
GROUP BY est.source_type;
```

- If the above read did not return results, display a message in place of the table, indicating “No Records Found”.
- When user clicks the ***Back*** button, show the **Reports** page.

Off-the-grid household dashboard

Abstract Code

- There will be six tables on this page:
  - The first table requires a read on Household which returns the state which has the most households without *public utilities* ($PublicUtilities) (as a whole number) (off-the-grid) and a count of the number of off-the-grid households in that state (as a whole number).

```
SELECT l.state as state, COUNT(*) as count_off_grid
FROM Household h
INNER JOIN Location l on l.postal_code=h.postal_code
WHERE NOT EXISTS
(SELECT * from PublicUtility pu
WHERE pu.household_email=h.email)
GROUP BY l.state
ORDER BY count_off_grid DESC
LIMIT 1;
```

  - The second table requires a read on PowerGenerator and a read on Household which returns a list of results of all households without *public utilities* ($PublicUtilities) (off-the-grid) and the average battery *storage capacity* ($StoragekWh) per battery as a whole number, rounded.

```
SELECT ROUND(AVG(pg.battery_storage_capacity), 0) as
avg_battery_storage_capacity
```

Table of Contents

```
FROM Household h
INNER JOIN PowerGenerator pg on pg.household_email=h.email
WHERE NOT EXISTS
(SELECT * from PublicUtility pu
WHERE pu.household_email=h.email);
```

- The third table requires a read on PowerGenerator and a read on Household, returning a list of results of the breakdown of *power generation type* ('$PowerGeneratorType') by percentage for all households without *public utilities* ($PublicUtilities) (off-the-grid) as decimal numbers, rounded to tenths.
  - If partial results are returned, use "0%" in place of missing values.

```
SELECT
      (SELECT
            GROUP_CONCAT(DISTINCT pg.type SEPARATOR ' & ')
            FROM PowerGenerator pg
            WHERE pg.household_email=h.email)
      AS household_power_generator_type,
      CONCAT(ROUND(((COUNT(*)/(SELECT COUNT(*)
      FROM Household h
      WHERE NOT EXISTS
      (SELECT * from PublicUtility pu
      WHERE pu.household_email=h.email)))*100), 1), '%') AS
percent
FROM Household h
WHERE NOT EXISTS
(SELECT * from PublicUtility pu
WHERE pu.household_email=h.email)
GROUP BY 1;
```

- The fourth table requires a read on Household returning a list of results of the breakdown of *home types* ('$HomeType') by percentage for all households without *public utilities* ($PublicUtilities) (off-the-grid) as decimal numbers, rounded to tenths.
  - All household types must be displayed even if that household type has no off-the-grid households.

```
SELECT ht.house_type,
CONCAT(ROUND((((SELECT COUNT(*)
      FROM Household h
      WHERE NOT EXISTS
      (SELECT * from PublicUtility pu
            WHERE pu.household_email=h.email)
            AND ht.house_type=h.type)/
      (SELECT COUNT(*)
            FROM Household h
            WHERE NOT EXISTS
                  (SELECT * from PublicUtility pu
                  WHERE pu.household_email=h.email)))*100), 1),
'%')
```

Table of Contents

```
AS percent_type
FROM (SELECT distinct(type) house_type FROM Household) ht;
```

- The fifth table requires a read on Appliance and a read on Household returning a list of results of the average *water heater tank size* (‘$ApplianceWaterHeaterTankSize’) for all households without *public utilities* (‘$PublicUtilities’) (off-the-grid) and for all households with *public utilities* (‘$PublicUtilities’) (on-the-grid). Both should be returned as a decimal number, rounded to tenths.

```
SELECT
ROUND(AVG(WHong.tank_size), 1) as
avg_water_heater_tank_size_ong,
ROUND(AVG(WHoffg.tank_size), 1) as
avg_water_heater_tank_size_offg
FROM Household h
LEFT JOIN
(SELECT h.email from Household h
INNER JOIN PublicUtility pu ON h.email=pu.household_email
WHERE pu.household_email IS NOT NULL) AS ong on
h.email=ong.email
LEFT JOIN
(SELECT h.email from Household h
WHERE NOT EXISTS
      (SELECT * from PublicUtility pu
      WHERE pu.household_email=h.email)) AS offg on
h.email=offg.email
LEFT JOIN WaterHeater WHong ON ong.email=WHong.household_email
LEFT JOIN WaterHeater WHoffg ON
offg.email=WHoffg.household_email;
```

- The sixth table requires a read on PowerGenerator returning a list of results, grouped by *appliance type* (‘$ApplianceType’), for the minimum, maximum, and average *BTU rating* (‘$ApplianceWaterHeaterBTU’) as whole numbers, rounded for all households without *public utilities* (‘$PublicUtilities’) (off-the-grid).
  - If partial results are returned, use zero in place of missing values.

```
WITH Appliance as
(SELECT 'Air Handler' as appliance_type, btu_rating,
household_email
FROM AirHandler
UNION ALL
SELECT 'Water Heater' as appliance_type, btu_rating,
household_email
FROM WaterHeater)
SELECT a.appliance_type,
ROUND(MIN(a.btu_rating), 0) as min_btu_rating,
ROUND(MAX(a.btu_rating), 0) as max_btu_rating,
ROUND(AVG(a.btu_rating), 0) as avg_btu_rating
FROM Household h
```

Table of Contents

```
INNER JOIN PowerGenerator pg on pg.household_email=h.email
INNER JOIN Appliance a on a.household_email=h.email
WHERE NOT EXISTS
(SELECT * from PublicUtility pu
WHERE pu.household_email=h.email)
GROUP BY a.appliance_type;
```

- If any of the above six read locks did not return results, display a message in place of the table, indicating “No Records Found”.
- When user clicks the ***Back*** button, show the **Reports** page.

Household averages by radius

Abstract Code

- Read on Location to populate a list of allowed postal codes.

```
SELECT postal_code
FROM Location;
```

- User enters *postal code* ('$PostalCode’) that’s within the allowed list of postal codes and *postal code search radius* ('$PostalCodeSearchRadius’) of allowable values of 0, 5, 10, 25, 50, 100, and 250 input fields.
    - If 0 is chosen for the *postal code search radius*, then we search in only the specified *postal code*.
- If data validation is successful for both *postal code* and *distance of postal code* input fields, then:
    - Read on Appliance, Household, PublicUtility, PowerGenerator, and PowerGenerator.
        - If any of the above read locks did not return results, display a message in place of the table, indicating “No Records Found”.
        - Else if results exist:
            - To compute the distance between the *postal codes*, use the haversine formula to calculate the straight-line distance between two points. The formula is noted below:

                $a = \sin^2(\Delta\varphi/2) + \cos\varphi1 \times \cos\varphi2 \times \sin^2(\Delta\lambda/2)$
                $c = 2 \times \text{atan2}(\sqrt{a}, \sqrt{(1-a)})$
                $d = R \times c$
                where φ is latitude, λ is longitude, R is earth’s radius (mean radius = 6,371km or approx. 3958.75 mi);
                - In Location, the latitude and longitude columns are expressed in degrees, which will be converted to radians for these calculations.
            - The query returns a list of the following data:
                - ‘$PostalCode’ from Location
                - ‘$PostalCodeSearchRadius’
                - The total number of households in the

Table of Contents

$PostalCodeSearchRadius from Household
    - For each *household type* ('$HomeType'):
        - Count of households from Household
            - If the count is 0 for a specific *household type*, show the *household type* anyway
        - Average *square footage* ($SquareFeet) as a whole number, rounded from Household
        - Average *heating temperature* ($ThermostatHeating) as a decimal number rounded to tenths from Household
        - Average *cooling temperature* ($ThermostatCooling) as a decimal number rounded to tenths from Household
        - List of *Public utilities* ($PublicUtilities) displayed in a single cell, separated by commas from PublicUtility
        - Count of "off-the-grid" homes from PublicUtility
        - Count of homes with *power generation* ($PowerGenerators) from PowerGenerator
        - Top 1 generation method for all households with power generation ($PowerGenerators) from PowerGenerator
        - Average monthly power generation per household ($PowerGenerators) as a whole number, rounded from PowerGenerator
        - Count of households with battery storage ($StoragekWh) from PowerGenerator
- Else *postal code* or *distance of postal* input fields are invalid, display **Household averages by radius** page, with an error message on the specific field.
- When user clicks the ***Back*** button, show the **Reports** page.

```
WITH distance_from AS
(SELECT * FROM (
  SELECT
      L.postal_code as from_postal_code,
        L.city AS from_city,
        L.longitude AS from_longitude,
        L.latitude AS from_latitude,
        loc.postal_code as to_postal_code,
        loc.city AS to_city,
        loc.longitude AS to_longitude,
        loc.latitude AS to_latitude,
        2 * 6335
          * asin(sqrt(
              power(sin((radians(loc.latitude) - radians(L.latitude)) / 2),
2)
              + cos(radians(L.latitude))
              * cos(radians(loc.latitude))
              * power(sin((radians(loc.longitude) - radians(L.longitude)) /
2), 2)
        ))
```

Table of Contents

```
        -- Haversine formula from
http://www.movable-type.co.uk/scripts/latlong.html
        as DISTANCE
    FROM
        Location L
    CROSS JOIN Location loc
  )
DIST
Where distance<='$PostalCodeSearchRadius' and from_postal_code='$PostalCode'
ORDER BY distance ASC)
SELECT '$PostalCode' as postal_code,
'$PostalCodeSearchRadius' as search_radius,
COUNT(h.email) as count_households,
SUM(CASE h.type WHEN 'Home' THEN 1 ELSE 0 END) as count_home,
SUM(CASE h.type WHEN 'Apartment' THEN 1 ELSE 0 END) as count_apartment,
SUM(CASE h.type WHEN 'Townhome' THEN 1 ELSE 0 END) as count_townhome,
SUM(CASE h.type WHEN 'Condominium' THEN 1 ELSE 0 END) as count_condo,
SUM(CASE h.type WHEN 'Modular Home' THEN 1 ELSE 0 END) as
count_modular_home,
SUM(CASE h.type WHEN 'Tiny House' THEN 1 ELSE 0 END) as count_tiny_house,
ROUND(AVG(h.square_footage), 1) as avg_sq_footage,
ROUND(AVG(h.thermostat_cooling), 1) as avg_cooling,
ROUND(AVG(h.thermostat_heating), 1) as avg_heating,
(GROUP_CONCAT((SELECT
      pu.name
      from PublicUtility pu
      where pu.household_email=h.email)
      SEPARATOR ', '))
      as utility_types,
(SELECT COUNT(*)
      FROM Household h
      INNER JOIN distance_from df on df.to_postal_code=h.postal_code
      WHERE NOT EXISTS
            (SELECT * from PublicUtility pu
            WHERE pu.household_email=h.email)) as count_off_grid,
(SELECT COUNT(*)
      FROM Household h
      INNER JOIN distance_from df on df.to_postal_code=h.postal_code
      WHERE EXISTS
            (SELECT * from PowerGenerator pg
            WHERE pg.household_email=h.email)) as count_power_generation,
(SELECT tt.generator_type from
      (SELECT pg.`type` as generator_type,
      COUNT(*)
      FROM PowerGenerator pg
      JOIN Household h on pg.household_email=h.email
      GROUP BY 1
```

Table of Contents

```
      ORDER BY 2 DESC
      LIMIT 1) tt) as top_generation_type,
(SELECT AVG((SELECT ROUND(SUM(pg2.avg_monthly_kwh)/COUNT(DISTINCT
pg2.household_email), 0)
      FROM PowerGenerator pg2
      WHERE pg2.household_email=h.email ))) as average_power_generation,
(SELECT COUNT(
  (SELECT COUNT(DISTINCT pg3.household_email)
      FROM PowerGenerator pg3
      WHERE pg3.household_email=h.email
      AND pg3.battery_storage_capacity IS NOT NULL)
)) as count_battery_storage
FROM Household h
INNER JOIN distance_from di on h.postal_code=di.to_postal_code;
```

Table of Contents