Table of Contents

Task Decomposition with Abstract Code

Main Menu

View Report/Query data

Enter Email Address

Enter Postal Code

Enter Phone Number

Enter House Info

Add Bathroom

View Bathrooms

Add Appliance

View Appliances

Generate Top 25 Manufacturers Report

Generate Manufacturer's Appliance List Report

Generate Manufacturer/Model Search Report

Generate Average TV Display Size by State Report

Generate State TV Display Drill Down Report

Generate Extra Fridge/Freezer Report

Generate Laundry Center Report

Generate Bathroom Stats Report

Generate Household Average by Radius Report

Main Menu

Abstract Code

- Upon:
    - Click ***Enter my household info*** link – jump to the **Enter Email Address** form.
    - Click ***View reports/query data*** link – jump to the **View Report/Query data** form.

View Report/Query data

Abstract Code

- After user clicks the ***View reports/query data*** link on the **Main Menu** form:
    - Upon:
        - Click ***Top 25 Manufacturers Report*** – Jump to **Generate Top 25 Manufacturers Report** form
        - Click ***Manufacturer/Model Search Report*** – Jump to **Generate Manufacturer/Model Search Report** form
        - Click ***Average TV Display Size by State Report*** – Jump to **Generate Average TV Display Size by State** form
        - Click ***Extra Fridge/Freezer Report*** – Jump to **Generate Extra Fridge/Freezer Report** form
        - Click ***Laundry Center Report*** – Jump to **Generate Laundry Center Report** form
        - Click ***Bathroom Statistics Report*** – Jump to **Generate Bathroom Statistics Report** form
        - Click ***Household Averages by Radius Report*** – Jump to Generate **Household Averages by Radius Report** form
        - Click ***Back to Main Menu*** button – Go to **Main Menu** form

Enter Email Address

Abstract Code

- After user clicks the ***Enter my household info*** link on the **Main Menu** form:
    - User enters household *email address* ($email_address) in "Please enter your email address:" input field.
    - ***Submit*** button is clicked:
        - If email format is valid:

```
SELECT EmailAddress FROM `Household` WHERE EmailAddress='$email_address;
```

- If user household email address is found:
    - Error message displayed on **Enter Email Address** form indicating that the email address entered is already associated with a household. Prompt user to use a different email address.
- Else:
    - Store $email_address as session variable $Email.
    - Go to **Enter Postal Code** form.

        - Else email address input field is invalid and error message indicating the expected format is displayed on **Enter Email Address** form.

Enter Postal Code

Abstract Code

- After user clicks ***Submit*** button on the **Enter Email Address** form:
    - User enters household *postal code* ($postal_code) in “Please enter your five digit postal code:” input field.
    - ***Submit*** button is clicked:
        - If data validation is successful for postal code:

```
SELECT PostalCode, City, State FROM Region WHERE PostalCode='$postal_code';
```

- If user household postal code is not found:
    - Error message displayed on **Enter Postal Code** form.
- Else:
    - Go to **Confirm Postal Code** page.
        - Display Region.PostalCode, Region.City, and Region.State
        - If user clicks ***YES*** button on **Confirm Postal Code** form:
            - Store $postal_code as session variable $PostalCode.
            - Go to **Enter Phone Number** form.
        - If user clicks ***NO*** button on **Confirm Postal Code** form:
            - Go back to **Enter Postal Code** form.

- Else postal code input field is invalid and error message is displayed indicating that the postal code did not match the expected format and the correct format on **Enter Postal Code** form.

Enter Phone Number

Abstract Code

- After user clicks ***Submit*** button on the **Enter Postal Code** form
    - If answer for "Would you like to enter a phone number" is *No*:
        - Hide enter phone number form
        - Set session variable $EnteredPhoneNumber = false
    - If answer for "Would you like to enter a phone number" is *Yes*:
        - Display enter phone number form.
        - Set session variable $EnteredPhoneNumber = true
        - User enters *area code* ($area_code) in "Area code:" input field and *phone number* ($number) in "Number:" input field
        - User selects 'home', 'mobile', 'work', or 'other' ($phone_type) from "Phone type:" drop down list.
    - When user clicks ***Next*** button:
        - If $EnteredPhoneNumber = true
            - If area code and phone number formats are valid:
                - Remove '-' from $number, if present

```
Select AreaCode, Number FROM PhoneNumber WHERE AreaCode='$area_code' AND Number='$number';
```

-
    -
        -
            -
                - If area code and phone number is found, display error on **Enter Phone Number** form indicating that the phone number is associated with another household and prompt the user to enter another phone number.
                - Else:
                    - Store $area_code as session variable $AreaCode.
                    - Store $number as session variable $Number.
                    - Store $phone_type as session variable $PhoneType.
            - Else: Display error on **Enter Phone Number** form indicating that the input did not satisfy the required format and prompt user to try again.
    - Go to **Enter House Info** form.

Enter House Info

Abstract Code

- After the user clicks the ***Next*** button on the **Phone Number Entry** form:
    - User selects 'House', 'apartment', 'townhome', 'condominium', or 'mobile home' ($home_type) from "Home type:" dropdown menu.
    - User enters information for *square footage* ($square_footage) in "Square Footage:" input field, and *occupants* ($occupants) in "Occupants:" input field, and *bedrooms* ($bedrooms) in "Bedrooms:" input field.
    - When the user clicks the ***Next*** button
        - If all input is valid:
            - Store $home_type as session variable $HomeType.
            - Store $square_footage as session variable $SquareFootage.
            - Store $occupants as session variable $Occupants.
            - Store $bedrooms as session variable $Bedrooms.
            - Go to the **Add Bathroom** form
        - Else
            - Display error on **Enter Household Info** form and indicating which fields did not match the required format and prompt user to correct those fields.

Add Bathroom

Abstract Code

- After the user clicks the ***Next*** button on the **Enter House Info** form or after the user clicks the ***Add Another Bathroom*** button on the **View Bathrooms** screen, they will be navigated to **Add Bathroom** form.
    - If user selects the ***Half*** tab under "Bathroom Type:"
        - Display "Name:", "Sinks:", "Commodes:", and "Bidets:" input fields
        - User optionally enters *bathroom name* ($name) in "Name:" input field
        - User enters *number of sinks* ($sinks) in "Sinks:" input field, *number of commodes* ($commodes) in "Commodes:" input field, and *number of bidets* ($bidets) in "Bidets:" input field.
        - When the user clicks the ***Add*** button:
            - If $sinks + $commodes + $bidets >= 1:
                - Create new Half Bathroom data structure to temporarily store data ($HalfBathStruct)
                - Increment session variable $NumBathrooms (initialized to 0) by 1
                - Store $name as $HalfBathStruct->Name.
                - Store $sinks as $HalfBathStruct->Sinks.
                - Store $commodes as $HalfBathStruct->Commodes.
                - Store $bidets as $HalfBathStruct->Bidets.
                - Store $HalfBathStruct in session associative array variable $Bathrooms[$NumBathrooms]
                - Go to the **View Bathrooms** form
            - Else: Display an error indicating that a Half bathroom must have at least one sink, commode, or bidet.
    - If user selects the ***Full*** tab under "Bathroom Type:"
        - Display "Sinks:", "Commodes:", "Bidets:", "Bathtubs:", "Showers:", and "Tub/showers:" input fields, and a "This bathroom is a primary bathroom" checkbox to indicate whether the bathroom is a primary bathroom.
        - If session variable $PrimaryEntered (initialized to false) == true:
            - a primary bathroom has already been added for this house, so disable the "This bathroom is a primary bathroom" checkbox.
        - Else:
            - No primary bathroom has been added yet, so enable the checkbox.
        - User enters *number of sinks* ($sinks) in "Sinks:" input field, *number of commodes* ($commodes) in "Commodes:" input field, *number of bidets* ($bidets) in "Bidets:" input field, *number of bathtubs* ($bathtubs) in "Bathtubs:" input field, *number of showers* ($showers) in "Showers:" input field, *number of tub/showers* ($tub_showers) in "Tub/showers:" input field, and checks or unchecks the "This bathroom is a primary bathroom" checkbox ($is_primary).
        - When the user clicks the ***Add*** button:
            - If $bathtubs + $showers + $tub_showers >= 1:

```
            o  Create new Full Bathroom data structure to temporarily store
               data ($FullBathStruct)
            o  Increment session variable $NumBathrooms (initialized to 0) by
               1
            o  Store $sinks as $FullBathStruct->Sinks.
            o  Store $commodes as session variable $FullBathStruct-
               >Commodes.
            o  Store $bidets as session variable $FullBathStruct->Bidets.
            o  Store $bathtubs as session variable $FullBathStruct->Bathtubs.
            o  Store $showers as session variable $FullBathStruct->Showers.
            o  Store $tub_showers as session variable $FullBathStruct-
               >TubShowers.
            o  Store $FullBathStruct in session associative array variable
               $Bathrooms[$NumBathrooms]
            o  If $is_primary == true:
                  ▪  $PrimaryEntered = true
            o  Go to the View Bathrooms form
        •  Else: Display error that a Full bathroom must have at least one bathtub,
           shower, or tub/shower.
    ▪  Else
        •  Remain on the Add Bathroom form and follow prompts to add missing
           bathroom components
```

View Bathrooms

Abstract Code

- After the user adds a bathroom on the **Add Bathroom** form:
  - Display a table with header row consisting of "Bathroom #", "Type", and "Primary" columns and table body populated in the following way:
    - Loop over bathroom structures in session associative array variable $Bathrooms for $i from 1 to $NumBathrooms:
      - Begin a new row
      - Display $i in the "Bathroom # column
      - Display the type of $Bathroom[$i] ("half" or "full") in the "Type" column
      - If type of $Bathroom[$i] is full bathroom and $Bathroom[$i]->IsPrimary == true
        - Display "Yes" in "Primary" column
      - Else:
        - Leave "Primary" column blank
  - If the user clicks ***Add Another Bathroom*** button
    - Go to **Add Bathroom** form to add another bathroom
  - When the user clicks the ***Next*** button
    - Go to the **Add Appliance** form

Add Appliance

Abstract Code

- After the user clicks the ***Next*** button on the View Bathrooms screen, or after the user clicks the ***Add Another Appliance*** button on the **View Appliances** screen, they will be navigated to **Add Appliance** form

```
SELECT ManufacturerName FROM Manufacturer;
```

  - Populate "Manufacturer" drop down with list of manufacturer names resulting from the query above
  - User selects "Refrigerator/Freezer", "Cooker", "Washer", "Dryer", and "TV" ($appliance_type) from the Appliance Type drop down menu.
  - User will select a manufacturer ($manufacturer) from the "Manufacturer:" dropdown menu.
  - User optionally enters model name ($model_name) in the "Model name:" input field.
  - After Appliance Type is chosen, a set of type-specific input field will be displayed, according to the following conditions:
    - If $appliance_type == "Refrigerator/Freezer":
      - User selects "Bottom freezer refrigerator", "French door refrigerator", "side-by-side refrigerator", "top freezer refrigerator", "chest freezer", or "upright freezer" ($fridge_type) from the "Refrigerator Type:" dropdown menu
      - When the user clicks the ***Add*** button:
        - Create new refrigerator data structure to temporarily store data ($RefrigeratorStruct)
        - Increment session variable $NumAppliances (initialized to 0) by 1
        - Store $appliance_type as $RefrigeratorStruct->ApplianceType
        - Store $model_name as $RefrigeratorStruct ->ModelName.
        - Store $manufacturer as $RefrigeratorStruct->Manufacturer.
        - Store $fridge_type as $RefrigeratorStruct->RefrigeratorType.
        - Store $RefrigeratorStruct in session associative array variable $Appliances[$NumAppliances]
        - Go to the **View Appliances** form
    - Else if $appliance_type == "Washer":
      - User selects "top" or "front" ($loading_type) from the "Loading Type:" dropdown menu
      - When the user clicks the ***Add*** button:
        - Create new refrigerator data structure to temporarily store data ($WasherStruct)
        - Increment session variable $NumAppliances (initialized to 0) by 1
        - Store $appliance_type as $WasherStruct->ApplianceType

- Store $model_name as $WasherStruct->ModelName.
- Store $manufacturer as $WasherStruct->Manufacturer.
- Store $loading_type as $WasherStruct->LoadingType.
- Store $WasherStruct in session associative array variable $Appliances[$NumAppliances]
- Go to the **<u>View Appliances</u>** form

- Else if $appliance_type == "Dryer":
  - User selects "gas", "electric", or "none" ($heat_source) from the "Loading Type:" dropdown menu
  - When the user clicks the ***Add*** button:
    - Create new refrigerator data structure to temporarily store data ($DryerStruct)
    - Increment session variable $NumAppliances (initialized to 0) by 1
    - Store $appliance_type as $DryerStruct->ApplianceType
    - Store $model_name as $DryerStruct ->ModelName.
    - Store $manufacturer as $DryerStruct->Manufacturer.
    - Store $heat_source as $DryerStruct->HeatSource.
    - Store $DryerStruct in session associative array variable $Appliances[$NumAppliances]
    - Go to the **<u>View Appliances</u>** form
- Else if $appliance_type == "TV":
  - Users selects "tube", "DLP", "plasma", "LCD", or "LED" ($display_type) from the "Display Type:" dropdown menu
  - User selects "480i", "576i", "720p", "1080i", "1080p", "1440p", "2160p (4K)", or "4320p (8K)" ($max_resolution) from the "Maximum resolution:" dropdown menu
  - User enters *display size* ($display_size) in the "Display Size:" input field
  - When the user clicks the ***Add*** button:
    - If input validation for $display_size passes:
    - Create new refrigerator data structure to temporarily store data ($TvStruct)
    - Increment session variable $NumAppliances (initialized to 0) by 1
    - Store $appliance_type as $TvStruct->ApplianceType
    - Store $model_name as $TvStruct ->ModelName.
    - Store $manufacturer as $TvStruct->Manufacturer.
    - Store $display_type as $TvStruct->DisplayType.
    - Store $display_size as $TvStruct->DisplaySize.
    - Store $max_resolution as $TvStruct->MaxResolution.
    - Store $TvStruct in session associative array variable $Appliances[$NumAppliances]
      - Go to the **<u>View Appliances</u>** form

            - Else: Display an error indicating that a format of display size input is incorrect and prompt the user to enter another value on the **Add Appliance** form.
    - Else if $appliance_type == “Cooker”:
        - User checks or unchecks the “Oven” checkbox ($is_oven)
        - User checks or unchecks any combination of “gas” ($oven_gas), “electric” ($oven_electric), and/or “microwave” ($oven_microwave) checkboxes in the “Oven Heat source:” section
        - User selects “conventional” or “convection” ($oven_type) from the “Oven Type:” dropdown menu
        - User checks or unchecks the “Cooktop” checkbox ($is_cooktop)
        - User selects “gas”, “electric”, “radiant electric”, or “induction” ($cooktop_type) from the “Cooktop Type:” dropdown menu.
        - When the user clicks the ***Add*** button:
            - Create new refrigerator data structure to temporarily store data ($CookerStruct)
            - Increment session variable $NumAppliances (initialized to 0) by 1
            - Store $appliance_type as $CookerStruct->ApplianceType
            - Store $model_name as $CookerStruct->ModelName.
            - Store $manufacturer as $CookerStruct->Manufacturer.
            - Store $is_oven as $CookerStruct->IsOven.
            - Store $oven_gas as $CookerStruct->OvenGas.
            - Store $oven_electric as $CookerStruct->OvenElectric.
            - Store $oven_microwave as $CookerStruct->OvenMicrowave.
            - Store $oven_type as $CookerStruct->OvenType.
            - Store $is_cooktop as $CookerStruct->IsCooktop.
            - Store $cooktop_type as $CookerStruct->CooktopHeatSource.
            - Store $CookerStruct in session associative array variable $Appliances[$NumAppliances]
            - Go to the **View Appliances** form

View Appliances

Abstract Code

- After the user adds an appliance on the **Add Appliance** form:
    - Display a table with header row consisting of “Appliance #”, “Type”, “Manufacturer, and “Model” columns and table body populated in the following way:
        - Loop over bathroom structures in session associative array variable $Appliancess for $i from 1 to $NumAppliances:
            - Begin a new row
            - Display $i in the “Appliance # column
            - Display the type of $Appliance [$i] (“Refrigerator/Freezer”, “Cooker”, “Washer”, “Dryer”, and “TV”) in the “Type” column
            - Display the $Appliance [$i]->Manufacturer in the “Manufacturer” column
            - Display the $Appliance [$i]->ModelName in the “Model” column
    - If the user clicks ***Add Another Appliance*** button
        - Go to **Add Appliance** form
    - When the user clicks the ***Next*** button
    - Insert Household information:

```
INSERT INTO Household(EmailAddress, PostalCode, HomeType, SquareFootage, Occupants, Bedrooms) VALUES ('$Email', '$PostalCode', '$HomeType', $SquareFootage, $Occupants, $Bedrooms);
```

- If $PhoneNumberEntered == true:

```
INSERT INTO PhoneNumber(EmailAddress, AreaCode, Number, PhoneType) VALUES ('$Email', '$AreaCode', '$Number', '$PhoneType');
```

- Loop over bathroom structures in session associative array variable $Bathrooms for $i from 1 to $NumBathrooms:
    - If type of $Bathroom[$i] is full bathroom

```
INSERT INTO FullBath(BathroomNumber, EmailAddress, Sinks, Bidets, Commodes, IsPrimary, TubShowerCount, ShowerCount, BathtubCount) VALUES($i,'$Email', $Bathrooms[$i]->Sinks, $Bathrooms[$i]->Bidets, $Bathrooms[$i]->Commodes, $Bathrooms[$i]->IsPrimary, $Bathrooms[$i]->TubShowers, $Bathrooms[$i]->Showers, $Bathrooms[$i]->Bathrooms);
```

- Else if $Bathroom[$i]->Name is not empty:

```
INSERT INTO HalfBath(BathroomNumber, EmailAddress, Sinks, Bidets, Commodes, Name) VALUES($i,'$Email', $Bathrooms[$i]->Sinks, $Bathrooms[$i]->Bidets, $Bathrooms[$i]->Commodes, '$Bathroom[$i]->Name');
```

```
                ▪ Else
```

```
INSERT INTO HalfBath(BathroomNumber, EmailAddress, Sinks, Bidets, Commodes) VALUES($i,'$Email', $Bathrooms[$i]->Sinks,
$Bathrooms[$i]->Bidets, $Bathrooms[$i]->Commodes);
```

```
            o Loop over appliances structures in session associative array variable $Appliancess for $i
              from 1 to $NumAppliances:
                ▪ If $Appliances[$i]->ApplianceType == "Refrigerator/Freezer":
                    • If $Appliances[$i]->ModelName not empty:
```

```
INSERT INTO Refrigerator (ApplianceNumber, EmailAddress, ManufacturerName, ModelName, RefrigeratorType)
VALUES($i,'$Email', '$Appliances[$i]->Manufacturer', '$Appliances[$i]->ModelName', '$Appliances[$i]->RefrigeratorType');
```

```
                    • Else:
```

```
INSERT INTO Refrigerator (ApplianceNumber, EmailAddress, ManufacturerName, RefrigeratorType) VALUES($i,'$Email',
'$Appliances[$i]->Manufacturer', '$Appliances[$i]->Refrigeratortype');
```

```
                ▪ Else if $Appliances[$i]->ApplianceType == "Washer":
                    • If $Appliances[$i]->ModelName not empty:
```

```
INSERT INTO Washer (ApplianceNumber, EmailAddress, ManufacturerName, ModelName, LoadingType) VALUES($i,'$Email',
'$Appliances[$i]->Manufacturer', '$Appliances[$i]->ModelName', '$Appliances[$i]->LoadingType');
```

```
                    • Else:
```

```
INSERT INTO Washer (ApplianceNumber, EmailAddress, ManufacturerName, LoadingType) VALUES($i,'$Email', '$Appliances[$i]-
>Manufacturer', '$Appliances[$i]->LoadingType');
```

```
                ▪ Else if $Appliances[$i]->ApplianceType == "Dryer":
                    • If $Appliances[$i]->ModelName not empty:
```

```
INSERT INTO Dryer (ApplianceNumber, EmailAddress, ManufacturerName, ModelName, HeatSource) VALUES($i,'$Email',
'$Appliances[$i]->Manufacturer', '$Appliances[$i]->ModelName', '$Appliances[$i]->HeatSource');
```

```
                    • Else:
```

```
INSERT INTO Dryer (ApplianceNumber, EmailAddress, ManufacturerName, HeatSource) VALUES($i,'$Email', '$Appliances[$i]-
>Manufacturer', '$Appliances[$i]->HeatSource');
```

- Else if $Appliances[$i]->ApplianceType == “TV”:
  - If $Appliances[$i]->ModelName not empty:

```
INSERT INTO Tv (ApplianceNumber, EmailAddress, ManufacturerName, ModelName, DisplayType, DisplaySize,
MaximumResolution) VALUES($i,'$Email', '$Appliances[$i]->Manufacturer', '$Appliances[$i]->ModelName', '$Appliances[$i]-
>DisplayType', $Appliances[$i]->DisplaySize, '$Appliances[$i]->MaximumResolution');
```

  - Else:

```
INSERT INTO Tv (ApplianceNumber, EmailAddress, ManufacturerName, DisplayType, DisplaySize, MaximumResolution)
VALUES($i,'$Email', '$Appliances[$i]->Manufacturer', '$Appliances[$i]->DisplayType', $Appliances[$i]->DisplaySize,
'$Appliances[$i]->MaximumResolution');
```

- Else if $Appliances[$i]->ApplianceType == “Cooker”:
  - If $Appliances[$i]->ModelName not empty:

```
INSERT INTO Cooker (ApplianceNumber, EmailAddress, ManufacturerName, ModelName) VALUES($i,'$Email', '$Appliances[$i]-
>Manufacturer', '$Appliances[$i]->ModelName');
```

  - Else:

```
INSERT INTO Cooker (ApplianceNumber, EmailAddress, ManufacturerName) VALUES($i,'$Email', '$Appliances[$i]-
>Manufacturer');
```

  - If $Appliances[$i]->IsOven == true:

```
INSERT INTO Oven (ApplianceNumber, EmailAddress, OvenType) VALUES($i,'$Email', '$Appliances[$i]->OvenType');
```

    - If $Appliances[$i]->OvenGas == true:

```
INSERT INTO OvenHeatSource (ApplianceNumber, EmailAddress, HeatSource) VALUES($i,'$Email', 'gas');
```

    - If $Appliances[$i]->OvenElectric == true:

```
INSERT INTO OvenHeatSource (ApplianceNumber, EmailAddress, HeatSource) VALUES($i,'$Email', 'electric');
```

    - If $Appliances[$i]->OvenMicroWave == true:

```
INSERT INTO OvenHeatSource (ApplianceNumber, EmailAddress, HeatSource) VALUES($i,'$Email', 'microwave');
```

  - If $Appliances[$i]->IsCooktop == true:

```
INSERT INTO CookTop (ApplianceNumber, EmailAddress, HeatSource) VALUES($i,'$Email', '$Appliances[$i]-
>CooktopHeatSource');
```

Generate Top 25 Manufacturers Report

Abstract Code

- When User selects ***Top 25 Manufacturers Report*** button from "**View Report/query data**" screen

```
WITH AllAppliances AS (
        SELECT ManufacturerName FROM Refrigerator
        UNION ALL
        SELECT ManufacturerName FROM Washer
        UNION ALL
        SELECT ManufacturerName FROM Dryer
        UNION ALL
        SELECT ManufacturerName FROM Tv
        UNION ALL
        SELECT ManufacturerName FROM Cooker
)
SELECT ManufacturerName, COUNT(AllAppliances. ManufacturerName) manCount
FROM AllAppliances GROUP BY AllAppliances.ManufacturerName ORDER BY manCount DESC
LIMIT 25;
```

  - Display the list of the top 25 manufacturers resulting from the above query, according to appliance count; list in descending order
- Display ***Drilldown Report*** button next to each manufacturer.
- If ***Drilldown Report*** is clicked: Go to **Generate Manufacturer's Appliance List Report**
- If ***Back to Main Menu*** button is clicked: Go to **Main Menu** screen

Generate Manufacturer's Appliance List Report

Abstract Code

- When user clicks ***Drilldown Report*** button for a manufacturer ($InputManufacturer) from **Generate Top 25 Manufacturers Report** screen

```
WITH AllAppliances AS (
        SELECT 'Refrigerator' AS appliancetype, ManufacturerName FROM Refrigerator
        UNION ALL
        SELECT 'Washer' AS appliancetype, ManufacturerName FROM Washer
        UNION ALL
        SELECT 'Dryer' AS appliancetype, ManufacturerName FROM Dryer
        UNION ALL
        SELECT 'TV' AS appliancetype, ManufacturerName FROM Tv
        UNION ALL
        SELECT 'Cooker' AS appliancetype, ManufacturerName FROM Cooker
)
SELECT appliancetype, COUNT(ManufacturerName)
FROM AllAppliances
WHERE ManufacturerName = '$InputManufacturer'
GROUP BY appliancetype;
```

  - Display data with the following requirements:
    - $InputManufacturer displayed at top of report
    - Below the $InputManufacturer, display data table with Appliance Type in the left column and Appliance Count in the right column.
  - If ***Back to Main Menu*** button is clicked: Go to **Main Menu** screen

Generate Manufacturer/Model Search Report

Abstract Code

- After user clicks ***Manufacturer/model Search Report*** button on **View Report/Query Data** screen
    - If the user enters a string ($SearchText) into the *search bar* and clicks the ***Search*** button:
        - Manufacturer names matching the search string with all of their models and

```
WITH AllAppliances AS (
        SELECT ManufacturerName, ModelName FROM Refrigerator
        UNION ALL
        SELECT ManufacturerName, ModelName FROM Washer
        UNION ALL
        SELECT ManufacturerName, ModelName FROM Dryer
        UNION ALL
        SELECT ManufacturerName, ModelName FROM Tv
        UNION ALL
        SELECT ManufacturerName, ModelName FROM Cooker
)

SELECT DISTINCT ManufacturerName, ModelName FROM AllAppliances
WHERE LOWER(ManufacturerName) LIKE LOWER('%$SearchText%')
OR LOWER(ModelName) LIKE LOWER('%$SearchText%')
ORDER BY ManufacturerName ASC;
```

model names matching the search string with their manufacturers from the above query will be displayed in tabular form, ordered by manufacturer name ascending and model name ascending.

        - The cells containing manufacturer or model names matching the search string will be highlighted with a light green background.
- If ***Back to Main Menu*** button is clicked: Go to **Main Menu** screen

Generate Average TV Display Size by State Report

Abstract Code

- After user clicks ***Average TV Display Size by State*** button on **View Report/Query Data** screen

```
SELECT R.State, ROUND(AVG(T.DisplaySize), 1) as AvgTVDiplaySize FROM Tv as T
INNER JOIN Household as H ON H.EmailAddress=T.EmailAddress
INNER JOIN Region as R ON H.PostalCode=R.PostalCode
GROUP BY R.State ORDER BY R.State ASC;
```

  - In tabular format, display each state and its corresponding average TV display size (as a decimal number rounded up to the tenths decimal point).
  - Next to each row of the table, display a link to generate a drilldown report for each state
  - If there are no households in a state with a TV, display “Not Applicable” in the average display size column and disable the ***Drilldown Report*** link for that state
  - If a ***Drilldown Report*** link is clicked,
    - Go to **Generate State TV Display Drill Down Report** screen
  - If ***Back to Main Menu*** button is clicked: Go to **Main Menu** screen

Generate State TV Display Drill Down Report

Abstract Code

- After user clicks ***Drilldown Report*** button next to a state ($State) on the **Generate Average TV Display Size by State Report** screen

```
SELECT T.DisplayType, T.MaximumResolution, ROUND(AVG(T.DisplaySize), 1) as AvgTVDiplaySize
FROM Tv as T
INNER JOIN Household as H ON H.EmailAddress = T.EmailAddress
INNER JOIN Region as R ON H.PostalCode=R.PostalCode
WHERE R.State='$State'
GROUP BY T.DisplayType, T.MaximumResolution ORDER BY  AvgTVDiplaySize DESC;
```

  - Display the $State with which the report is associated.
  - In tabular format, display all screen types available in that $State, the maximum resolution for each type, and the average display size (as a decimal number rounded up to the tenths decimal point) for each type.
  - If ***Back to Main Menu*** button is clicked: Go to **Main Menu** screen

Generate Extra Fridge/Freezer Report

Abstract Code

- After user clicks **Extra Fridge/Freezer Report** button on **View Report/Query Data** screen

```
SELECT COUNT(RF.EmailAddress) AS HouseholdsCount
FROM (SELECT EmailAddress, count(ApplianceNumber) AS ApplianceCount FROM
Refrigerator GROUP BY EmailAddress) AS RF
WHERE RF.ApplianceCount > 1;
```

  - Display number of households with more than one fridge or freezer.
  - In tabular format, display the top ten states (states with the highest number of fridge/freezers) with columns for: the state; the count of households with multiple fridge/freezers in that state, sorted by household count descending; the percentage of households with multiple fridge/freezers in that state with chest freezers; the percentage of households with multiple fridge/freezers in that state with an upright freezer; the percentage of households with multiple fridge/freezers in that state with something else (SQL query next page).

```
WITH Top10StateHouseholdCounts AS(
SELECT RF.State, COUNT(RF.EmailAddress) AS HouseholdsCount, RF.ApplianceCount
        FROM (SELECT R.State, RF.EmailAddress, count(ApplianceNumber) AS ApplianceCount
                FROM Refrigerator AS RF
                INNER JOIN Household as H ON H.EmailAddress = RF.EmailAddress
                INNER JOIN Region as R ON H.PostalCode = R.PostalCode
                GROUP BY RF.EmailAddress) AS RF
WHERE RF.ApplianceCount > 1
GROUP BY RF.State, RF.ApplianceCount
ORDER BY HouseholdsCount DESC LIMIT 10),

Top10StatesChestRefrigerators AS(
SELECT R.State, COUNT(RF.ApplianceNumber) AS ApplianceCount FROM Refrigerator AS RF
INNER JOIN Household as H ON H.EmailAddress = RF.EmailAddress
INNER JOIN Region as R ON H.PostalCode = R.PostalCode
WHERE RF.RefrigeratorType='CHEST' AND RF.EmailAddress in
        (SELECT RF.EmailAddress FROM
                (SELECT R.State, RF.EmailAddress, count(ApplianceNumber) AS ApplianceCount
                        FROM Refrigerator AS RF
                        INNER JOIN Household as H ON H.EmailAddress = RF.EmailAddress
                        INNER JOIN Region as R ON H.PostalCode = R.PostalCode
                        GROUP BY RF.EmailAddress) AS RF
                WHERE RF.ApplianceCount > 1
                GROUP BY RF.State, RF.EmailAddress
                ORDER BY Count(RF.EmailAddress) DESC)
GROUP BY RF.EmailAddress
LIMIT 10),

Top10StatesUprightRefrigerators AS(
SELECT R.State, COUNT(RF.ApplianceNumber) AS ApplianceCount FROM Refrigerator AS RF
INNER JOIN Household as H ON H.EmailAddress = RF.EmailAddress
INNER JOIN Region as R ON H.PostalCode = R.PostalCode
WHERE RF.RefrigeratorType='UPRIGHT' AND RF.EmailAddress in
        (SELECT RF.EmailAddress FROM
                (SELECT R.State, RF.EmailAddress, count(ApplianceNumber) AS ApplianceCount FROM Refrigerator AS RF
                                INNER JOIN Household as H ON H.EmailAddress = RF.EmailAddress
                                INNER JOIN Region as R ON H.PostalCode = R.PostalCode
                                GROUP BY RF.EmailAddress) AS RF
                        WHERE RF.ApplianceCount > 1
                        GROUP BY RF.State, RF.EmailAddress
                        ORDER BY Count(RF.EmailAddress) DESC)
GROUP BY RF.EmailAddress
LIMIT 10),

Top10StatesOtherRefrigerators AS(
SELECT R.State, COUNT(RF.ApplianceNumber) AS ApplianceCount FROM Refrigerator AS RF
INNER JOIN Household as H ON H.EmailAddress = RF.EmailAddress
INNER JOIN Region as R ON H.PostalCode = R.PostalCode
WHERE RF.RefrigeratorType NOT IN ('CHEST','UPRIGHT') AND RF.EmailAddress in
        (SELECT RF.EmailAddress FROM
                (SELECT R.State, RF.EmailAddress, count(ApplianceNumber) AS ApplianceCount FROM Refrigerator AS RF
                                INNER JOIN Household as H ON H.EmailAddress = RF.EmailAddress
                                INNER JOIN Region as R ON H.PostalCode = R.PostalCode
                                GROUP BY RF.EmailAddress) AS RF
                        WHERE RF.ApplianceCount > 1
                        GROUP BY RF.State, RF.EmailAddress
                        ORDER BY Count(RF.EmailAddress) DESC)
GROUP BY RF.EmailAddress
LIMIT 10)

-- Final query for fridge/freezer report.
Select HC.State, HC.HouseholdsCount,
COALESCE(ROUND(ChestInfo.ApplianceCount * 100 / HC.ApplianceCount,0),0) AS HouseholdsWithChestFreezerTypePercentage,
COALESCE(ROUND(UprightInfo.ApplianceCount * 100 / HC.ApplianceCount,0),0) AS
HouseholdsWithUprightFreezerTypePercentage,
COALESCE(ROUND(OtherFridgeInfo.ApplianceCount * 100 / HC.ApplianceCount ,0),0) AS HouseholdsWithOtherTypesPercentage
FROM Top10StateHouseholdCounts AS HC
LEFT JOIN Top10StatesChestRefrigerators AS ChestInfo ON HC.State = ChestInfo.State
LEFT JOIN Top10StatesUprightRefrigerators AS UprightInfo ON HC.State = UprightInfo.State
LEFT JOIN Top10StatesOtherRefrigerators AS OtherFridgeInfo ON HC.State = OtherFridgeInfo.State;
```

  - If ***Back to Main Menu*** button is clicked: Go to **Main Menu** screen

Generate Laundry Center Report

Abstract Code

- After user clicks **Generate Laundry Center Report** button on **View Report/Query Data** screen
  - Find most common washer type and dryer heat source used by household per state.

```
WITH MostCommonWasherType AS(
 SELECT State, LoadingType
 FROM (
  SELECT State, LoadingType, ROW_NUMBER() OVER(PARTITION BY State ORDER BY LoadingTypeCount desc) as RowNum
  FROM (
   SELECT r.State as State, w.LoadingType as LoadingType, COUNT(w. LoadingType) as LoadingTypeCount
   FROM Household h
   INNER JOIN Region r ON r.PostalCode = h.PostalCode
   INNER JOIN Washer w ON w.EmailAddress = h.EmailAddress
   GROUP BY w.LoadingType, r.State
  ) AS t
 ) AS a
 WHERE RowNum = 1),

MostCommonDryerHeatSource AS(
 SELECT State, HeatSource
 FROM (
  SELECT State, HeatSource, ROW_NUMBER() OVER(PARTITION BY State ORDER BY DryerHeatSourceCount desc) as RowNum
  FROM (
   SELECT r.State as State, d.HeatSource as HeatSource, COUNT(d.HeatSource) as DryerHeatSourceCount
   FROM Household h
   INNER JOIN Region r ON r.PostalCode = h.PostalCode
   INNER JOIN Dryer d ON d.EmailAddress = h.EmailAddress
   GROUP BY d.HeatSource, r.State
  ) AS t
 ) AS a
 WHERE RowNum = 1)

 SELECT w.State, w.LoadingType, d.HeatSource
 FROM MostCommonWasherType w
 INNER JOIN MostCommonDryerHeatSource d ON d.State = w.State
```

- Display in a table with three columns: state, most common washer type, & most common dryer heat source.

    - Find every household including the house count with a washer machine but does not have a dryer.

```
SELECT r.State, COUNT(w.EmailAddress) as OnlyWasherCount
FROM Washer w
INNER JOIN Household h ON h.EmailAddress = w.EmailAddress
INNER JOIN Region r ON r.PostalCode = h.PostalCode
WHERE w.EmailAddress NOT IN (SELECT EmailAddress FROM Dryer)
GROUP BY r.State
```

        - Display in a table with two columns: state, & household count. Table should be ordered by household count descending.

- If ***Back to Main Menu*** button is clicked: Go to **Main Menu** screen

Generate Bathroom Stats Report

Abstract Code

- After user clicks **Bathroom Stats Report** button on **View Report/Query Data** screen
  - Find and display the minimum, average, and maximum count of all bathrooms, commodes, sinks, and bidets per household.

```
WITH AllBathrooms AS (
 SELECT EmailAddress, Sinks, Bidets, Commodes FROM HalfBath
 UNION ALL
 SELECT EmailAddress, Sinks, Bidets, Commodes FROM FullBath
)
SELECT
 MIN(BathroomCount) AS MinNumBathrooms,
 ROUND(AVG(BathroomCount), 1) AS AvgNumBathrooms,
 MAX(BathroomCount) AS MaxNumBathrooms,
 MIN(CommodesCount) AS MinNumCommodes,
 ROUND(AVG(CommodesCount), 1) AS AvgNumCommodes,
 MAX(CommodesCount) AS MaxNumCommodes,
 MIN(SinksCount) AS MinNumSinks,
 ROUND(AVG(SinksCount), 1) AS AvgNumSinks,
 MAX(SinksCount) AS MaxNumSinks,
 MIN(BidetsCount) AS MinNumBidets,
 ROUND(AVG(BidetsCount), 1) AS AvgNumBidets,
 MAX(BidetsCount) AS MaxNumBidets
FROM (
 SELECT h.EmailAddress,
 COUNT(b.EmailAddress) AS BathroomCount,
 COALESCE(SUM(b.Commodes),0) AS CommodesCount,
 COALESCE(SUM(b.Sinks),0) AS SinksCount,
 COALESCE(SUM(b.Bidets),0) AS BidetsCount
   FROM AllBathrooms b
   RIGHT JOIN Household h ON h.EmailAddress = b.EmailAddress
   GROUP BY h.EmailAddress
  ) AS t;
```

  - Find and display the minimum, average, and maximum count of half bathrooms per household.

```
SELECT
 MIN(BathroomCount) AS MinNumHalfBathrooms,
 ROUND(AVG(BathroomCount), 1) AS AvgNumHalfBathrooms,
 MAX(BathroomCount) AS MaxNumHalfBathrooms
FROM (
 SELECT h.EmailAddress, COUNT(b.EmailAddress) AS BathroomCount
 FROM HalfBath b
 RIGHT JOIN Household h ON h.EmailAddress = b.EmailAddress
 GROUP BY h.EmailAddress
) as t;
```

- Find and display the minimum, average, and maximum count of full bathrooms, bathtubs, showers, and tub/showers per household.

```
SELECT
  MIN(BathroomCount) AS MinNumFullBathrooms,
  ROUND(AVG(BathroomCount), 1) AS AvgNumFullBathrooms,
  MAX(BathroomCount) AS MaxNumFullBathrooms,
  MIN(TubShowerCount) AS MinNumTubShowers,
  ROUND(AVG(TubShowerCount), 1) AS AvgNumTubShowers,
  MAX(TubShowerCount) AS MaxNumTubShowers,
  MIN(ShowerCount) AS MinNumShowers,
  ROUND(AVG(ShowerCount), 1) AS AvgNumShowers,
  MAX(ShowerCount) AS MaxNumShowers,
  MIN(BathtubCount) AS MinNumBathtubs,
  ROUND(AVG(BathtubCount), 1) AS AvgNumBathtubs,
  MAX(BathtubCount) AS MaxNumBathtubs
FROM (
  SELECT h.EmailAddress,
  COUNT(b.EmailAddress) AS BathroomCount,
  COALESCE(SUM(TubShowerCount),0) AS TubShowerCount,
  COALESCE(SUM(ShowerCount),0) AS ShowerCount,
  COALESCE(SUM(BathtubCount),0) AS BathtubCount
     FROM FullBath b
     RIGHT JOIN Household h ON h.EmailAddress = b.EmailAddress
     GROUP BY h.EmailAddress
) AS t;
```

- Find and display and postal code with most bidets and total.

```
WITH AllBathrooms AS (
  SELECT EmailAddress, Sinks, Bidets, Commodes FROM HalfBath
  UNION ALL
  SELECT EmailAddress, Sinks, Bidets, Commodes FROM FullBath
)
SELECT PostalCode, NumBidets FROM(
  SELECT r.PostalCode AS PostalCode, SUM(Bidets) as NumBidets
  FROM AllBathrooms b
  INNER JOIN Household h ON h.EmailAddress = b.EmailAddress
  INNER JOIN Region r ON r.PostalCode = h.PostalCode
  GROUP BY r.PostalCode) AS t
ORDER BY NumBidets DESC
LIMIT 1;
```

- Find and display and postal code with most bidets and total.

```
WITH AllBathrooms AS (
  SELECT EmailAddress, Sinks, Bidets, Commodes FROM HalfBath
  UNION ALL
  SELECT EmailAddress, Sinks, Bidets, Commodes FROM FullBath
)
SELECT State, NumBidets FROM(
  SELECT r.State AS State, SUM(Bidets) as NumBidets
  FROM AllBathrooms b
  INNER JOIN Household h ON h.EmailAddress = b.EmailAddress
  INNER JOIN Region r ON r.PostalCode = h.PostalCode
  GROUP BY r.State
   ) AS t
 ORDER BY NumBidets DESC
 LIMIT 1;
```

Generate Household Average by Radius Report

Abstract Code

- After user clicks **Bathroom Stats Report** button on **View Report/Query Data** screen
    - When User enters *postal code* ($PostalCodeInput) and *radius* ($RadiusInput) and clicks the ***search*** button: check input validity

```
SELECT postalcode FROM Region WHERE postalcode = '$PostalCodeInput'
```

- If input is not valid (i.e., postalcode does not exist in database)
    - Display error message
- If Input is valid
    - Use $PostalCodeInput to lookup latitude and longitude of the corresponding location
    - Use $RadiusInput and latitude and longitude of $PostalCodeInput in the Haversine formula to compute the distance between $PostalCodeInput and all postal codes in the database, returning all postal codes that fall within the input radius

```
WITH PostalCodesInRadius AS (
  WITH InputRegion AS (
    SELECT PostalCode, City, Latitude AS inputlat, Longitude AS inputlong
    FROM Region
    WHERE PostalCode = '$PostalCodeInput')

  SELECT City, PostalCode, d
  FROM(
    SELECT City, PostalCode, 3598.75 * c AS d
      FROM(
      SELECT City, PostalCode, 2*atan2(SQRT(a),SQRT(1-a)) AS c
        FROM(
        SELECT LatLongDelta.City AS City, LatLongDelta.PostalCode AS PostalCode,
          (POWER(SIN((deltalat)/2),2) +
          COS(InputRegion.inputlat*PI()/180)*
          COS(radlat)*
          POWER(SIN((deltalong)/2),2)) AS a
          FROM InputRegion, (
            SELECT Region.City AS City,
            Region.PostalCode AS PostalCode,
            (Region.Latitude*PI()/180) AS radlat,
            (Region.Latitude-InputRegion.inputlat)*PI()/180 AS deltalat,
            (Region.Longitude*PI()/180) AS radlong,
            (Region.Longitude-InputRegion.inputlong)*PI()/180 AS deltalong
            FROM Region, InputRegion
            ) LatLongDelta
          ) HaversineC
        ) HaversineD
      ) HaversineResult
  WHERE d <= $RadiusInput ORDER BY d
  ),

HouseholdQuery AS (
  SELECT COUNT(EmailAddress) housecount, CAST(SUM(Occupants) AS DECIMAL) allOccupants, CAST(AVG(Occupants) AS DECIMAL)
occupantavg, AVG(Bedrooms) bedroomavg
  FROM Household, PostalCodesInRadius
  WHERE Household.PostalCode IN (PostalCodesInRadius.PostalCode)),

BathroomQuery AS (
            WITH BathroomUnion AS (
            SELECT EmailAddress, Commodes FROM FullBath
            UNION ALL
            SELECT EmailAddress, Commodes FROM HalfBath)
  SELECT COUNT(BathroomUnion.EmailAddress) AS bathroomCount, SUM(Commodes) AS commodeCount
  FROM BathroomUnion
  JOIN Household ON Household.EmailAddress = BathroomUnion.EmailAddress,
  PostalCodesInRadius
  WHERE Household.PostalCode IN (PostalCodesInRadius.PostalCode)),

ApplianceCount AS (
  WITH ApplianceUnion AS (
    SELECT EmailAddress FROM Refrigerator
    UNION ALL
    SELECT EmailAddress FROM Washer
    UNION ALL
    SELECT EmailAddress FROM Dryer
    UNION ALL
    SELECT EmailAddress FROM Cooker
    UNION ALL
    SELECT EmailAddress FROM Tv)
  SELECT COUNT(ApplianceUnion.EmailAddress) AS appliancecount
  FROM ApplianceUnion
  JOIN Household ON Household.EmailAddress = ApplianceUnion.EmailAddress,
  PostalCodesInRadius
  WHERE Household.PostalCode IN (PostalCodesInRadius.PostalCode)),

CookerHeatSource AS (
            WITH CookerUnion AS (
            SELECT EmailAddress, HeatSource FROM Cooktop
            UNION ALL
            SELECT EmailAddress, HeatSource FROM OvenHeatSource)
  SELECT HeatSource, COUNT(HeatSource) AS heatSourceCount
  FROM CookerUnion
  JOIN Household ON Household.EmailAddress = CookerUnion.EmailAddress,
  PostalCodesInRadius
  WHERE Household.PostalCode IN (PostalCodesInRadius.PostalCode)
  GROUP BY HeatSource
  ORDER BY heatSourceCount
  LIMIT 1),

DryerHeatSource AS (
  SELECT HeatSource, COUNT(HeatSource) AS heatSourceCount
  FROM Dryer
  JOIN Household ON Household.EmailAddress = Dryer.EmailAddress,
  PostalCodesInRadius
  WHERE Household.PostalCode IN (PostalCodesInRadius.PostalCode)
  GROUP BY HeatSource
  ORDER BY heatSourceCount
  LIMIT 1)

SELECT '$PostalCodeInput ' AS postalCode, $RadiusInput AS searchRadius,
ROUND(BathroomQuery.bathroomCount/HouseholdQuery.houseCount,1) AS bathroomsPerHousehold,
ROUND(HouseholdQuery.bedroomAvg, 1) AS bedroomsPerHousehold,
ROUND(HouseholdQuery.occupantAvg) AS occupantsPerHousehold,
CONCAT('1:',CAST(ROUND((HouseholdQuery.allOccupants/BathroomQuery.commodecount),2) AS CHAR(50))) AS commodesPerOccupant,
ROUND(ApplianceCount.appliancecount/HouseholdQuery.housecount,1) AS appliancesPerHousehold,
CookerHeatSource.HeatSource AS CookerHeatSource,
DryerHeatSource.HeatSource AS DryerHeatSource
FROM HouseholdQuery, BathroomQuery, ApplianceCount, CookerHeatSource, DryerHeatSource
```

- Calculate and/or display postal code; search radius; average bathroom count per household; average bedroom count per household; average occupant count per household; ratio of commodes to occupants per household; average number of appliances per household; and most common heat source per household.

  - If ***Back to Main Menu*** button is clicked: Go to **Main Menu** screen