Table of Contents:

Data Types:

Category

Attribute	Data type	Nullable
Name	String	Not Null

City

Attribute	Data type	Nullable
Name	String	Not Null
State	String	Not Null
Population	Integer	Not Null

DateInstance

Attribute	Data type	Nullable
Occurrence	Date	Not Null

GoesOnSale

Attribute	Data type	Nullable
SalePrice	Float	Not Null

Holiday

Attribute	Data type	Nullable
Name	String	Not Null

Manager

Attribute	Data type	Nullable
EmailAddress	String	Not Null
Name	String	Not Null

Manufacturer

Attribute	Data type	Nullable
Name	String	Not Null
MaximumDiscount	Float	Null

Product

Attribute	Data type	Nullable
PID	Integer	Not Null
Name	String	Not Null
Price	Float	Not Null

Store

Attribute	Data type	Nullable
StoreNumber	Integer	Not Null
PhoneNumber	String	Not Null
StreetAddress	String	Not Null

StoreSellsProduct

Attribute	Data type	Nullable
Quantity	Integer	Not Null

Business Logic Constraints:

S&E Data Warehouse Sale Price[1]

- Sale prices may not be higher than retail prices.
- If a product goes on sale, it is on sale at the same price in all stores.
- A maximum discount of 0% means the product cannot be placed on sale.

S&E Data Warehouse Manager

- A manager who has become inactive may not be deleted from the system until they have been unassigned from all stores.

S&E Product

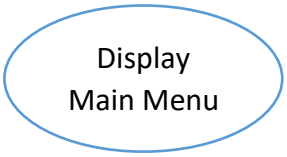
- If a product is on sale for multiple days in a row, then a record is stored in the data warehouse for each day of the sale.
- The retail price is in effect unless there is a sale.

[1] There seemed to be different interpretations supplied on Piazza regarding the mention that "[e]ven if a maximum discount is not specified by the manufacturer, as a general rule of S&E, no product can be discounted more than 90% of retail. Depending on how this is interpreted, it might be considered a business logic constraint or not. We have chosen to follow one of the latest instructor postings on Piazza that stated that this is not asking us to do anything if the rule is not followed, so we have not counted it as a constraint.

Display Main Menu

Task Decomp

Lock Types: Lookup **Store**.*PID*, **Manufacturer**.*Name*, and **Manager**.*EmailAddress*, all are Read-only.
Number of Locks: No locks needed.
Enabling Conditions: Trigger by successful login.
Frequency: Every successful login, but not critical.
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.



Abstract Code

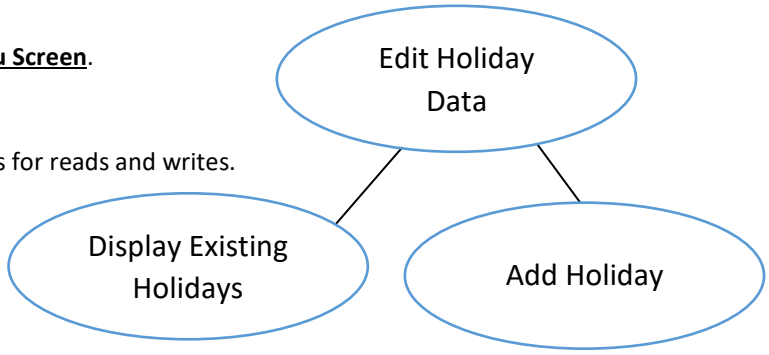
(Called upon successful login and upon ***Return to Main Menu*** button clicks.)

- Query database for statistics (counts of **Store**s, using **Store**.*StoreNumber*, **Manufacturer**s, using **Manufacturer**.*Name*, total managers, using **Manager**.*EmailAddress*, and active managers, using distinct **Manager**s who **Manages** any **Store**s) and populate statistics area of **Main Menu Screen** with those statistics.
- Show maintenance screen links (***Holiday Data***, ***Manager Profile***, ***Store Assignment***, and ***City Population***) on **Main Menu Screen.**
- Show report screen links (***Manufacturers' Product Report***, ***Category Report***, ***Actual/Predicted GPS Report***, ***Store Revenue Report***, ***GH Day AC Report***, ***State Volume by Category Report***, and ***Revenue by Population Report***) on **Main Menu Screen**.
- Upon:
    - Click ***Holiday Data*** link: Jump to the Edit Holiday Data task.
    - Click ***Manager Profile*** link: Jump to the Edit Manager Profile task.
    - Click ***Store Assignment*** link: Jump to the Assign Stores task.
    - Click ***City Population*** link: Jump to the Update City Population task.
    - Click ***Manufacturers' Product Report*** link: Jump to the Display Manufacturer Summary task.
    - Click ***Category Report*** link: Jump to the Display Category Summary task.
    - Click ***Actual/Predicted GPS Report*** link: Jump to the Display Actual vs Predicted GPS task.
    - Click ***Store Revenue Report*** link: Jump to the Display Store Revenue by Year and State task.
    - Click ***GH Day AC Report*** link: Jump to the Display Air Conditioners on Groundhog Day task.
    - Click ***State Volume by Category Report*** link: Jump to the Display State Volume by Category Report task.
    - Click ***Revenue by Population Report*** link: Jump to the Display Annual Average Revenue by City Population task.

Edit Holiday Data

Task Decomp

Lock Types: Lookup **Holiday**.*Occurrence* and **Holiday**.*Name* (read-only upon display). Write **Holiday**.*Occurrence* and **Holiday**.*Name* (write upon Holiday add).
Number of Locks: No locks needed.
Enabling Conditions: Trigger by click from **Main Menu Screen**.
Frequency: Low. Writes will be fewer than reads.
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is needed to separate subtasks for reads and writes.



Abstract Code

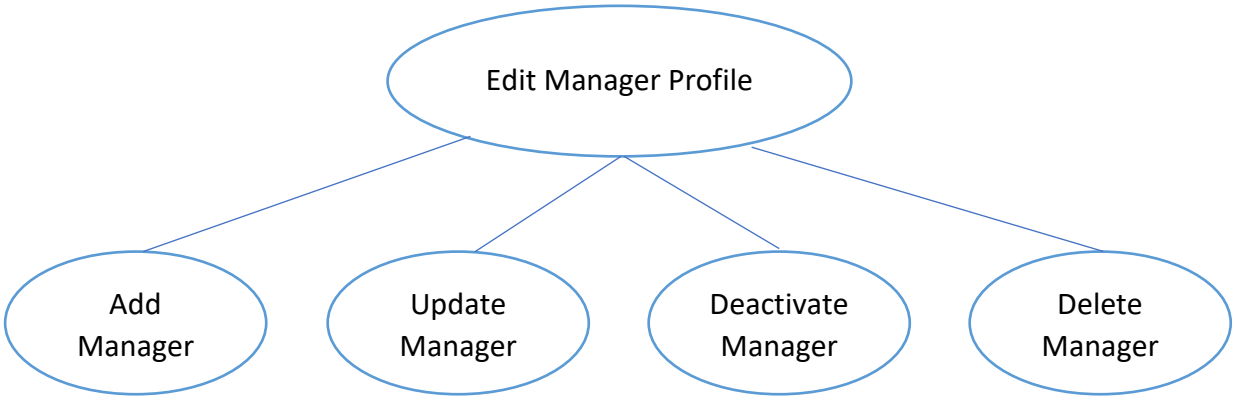
(Called from the **Main Menu Screen** and below)

- Clear all input fields
- Query database for current **Holiday**s (**Holiday**.*Occurrence* and **Holiday**.*Name*) and populate existing Holidays area of **Holiday Data Screen** with those **Holiday**s
- User will fill in **DATE** and **HOLIDAY NAME** input fields.
- Upon:
  - Click ***Add Holiday*** button:
    - Read **DATE** and **HOLIDAY NAME** input fields from Add section of **Holiday Data Screen**.
    - If Date is valid and does not already exist as a **Holiday**.*Occurrence* value:
      - Insert new **Holiday** instance with those values, then clear any success/error messages, display a success message, and call the Display Holiday Data task.
    - Otherwise, display an appropriate error message.
  - Click ***Return to Main Menu*** button: Call the Display Main Menu task.

Edit Manager Profile

Task Decomp

Lock Types: 4 write locks, one for add manager, one for update manager, one for deactivate manager, and one for delete manager
Number of Locks: 4
Enabling Conditions: Triggered from **Main Menu Screen**
Frequency: Low-
Consistency (ACID): Not critical, even if the **Manager** or **Store** is being edited by the user in a parallel thread.
Subtasks: 4 subtasks, shown and detailed below. Mother task: Edit Manager Profile is a mother task for all the **Manager**-related subtasks. Order: Not required.



Abstract Code

(Called from the **Main Menu Screen** and below)

- Display the **Manager Profiles Screen**.
- Clear all input fields and success/error messages
- Display list of all **Manager**s' names (**Manager**.*Name*) and email addresses (**Manager**.*EmailAddress*) in area provided on **Manager Profiles Screen**.
- Upon:
  - ***Add Manager*** button click: Read the **NAME** and **EMAIL ADDRESS** input fields on the **Manager Profiles Screen**. If both fields are non-empty, and the email is a valid email address that does not exist in any existing **Manager** instance, insert a new manager with those attribute values (**Manager**.*Name* and **Manager**.*EmailAddress*), display a success message, then call the Edit Manager Profile task. Otherwise, display an appropriate error message.
  - ***Update Manager*** button click: Read the **NAME** and **EMAIL ADDRESS** input fields on the **Manager Profiles Screen**. If both fields are non-empty and the email is a valid email address, find the **Manager** instance identified by that email address (**Manager**.*EmailAddress*). If no such instance exists, display an error message. Otherwise, update the **Manager** instance identified by the email address to hold the **Manager**.*Name* given by the **NAME** input field, display a success message, and then call the Edit Manager Profile task.
  - ***Deactivate Manager*** button click:
    - Clear the output area and any success/error messages on the **Manager Profiles Screen**.
    - Read the **EMAIL ADDRESS** input field on the **Manager Profiles Screen.**

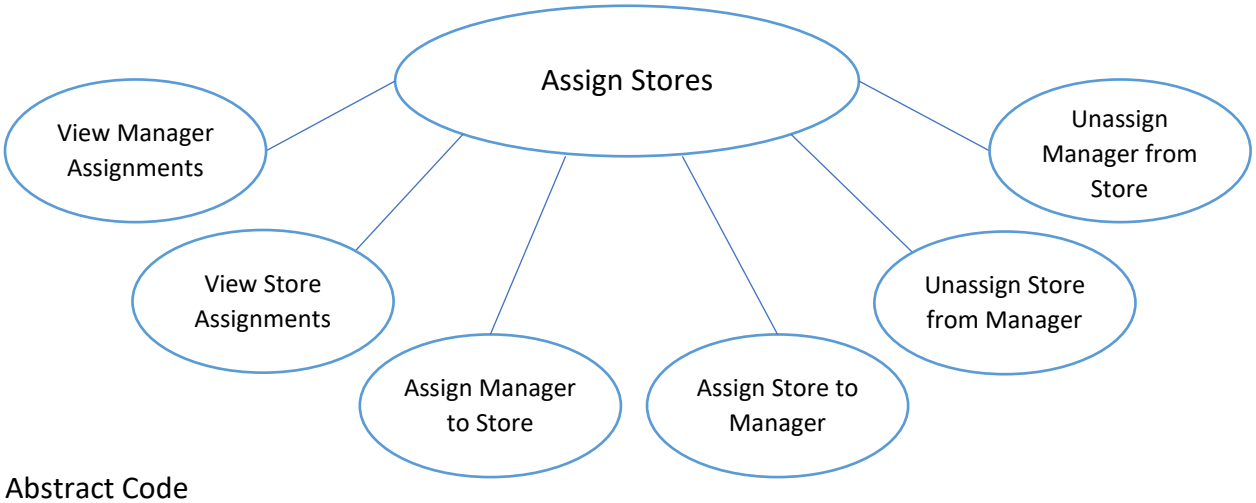
- If the field is non-empty, check to see if there is a **Manager** instance with that **Manager**.*EmailAddress*.
  - If so, check to see if that **Manager** (**Manager**.*EmailAddress*) and any **Store** (**Store**.*StoreNumber*) are already related by the **Manages** relationship.
    - If so, delete all **Manages** instances for that **Manager**.*EmailAddress* and display a success message.
    - Otherwise, write an appropriate error message ("already deactivated") on the **Manager Profiles Screen**.
- Otherwise, write an appropriate error message.

- ***Delete Manager*** button click:
  - Clear the output area and any success/error messages on the **Manager Profiles Screen**.
  - Read the **EMAIL ADDRESS** input field on the **Manager Profiles Screen**.
    - If the field is non-empty, check to see if there is a **Manager** instance with that **Manager**.*EmailAddress*.
      - If so, check to see if that **Manager** (**Manager**.*EmailAddress*) and any **Store** (**Store**.*StoreNumber*) are already related by the **Manages** relationship.
        - If not, delete the **Manager** instance with key **Manager**.*EmailAddress* and display a success message.
        - Otherwise, display an appropriate error message ("must first be deactivated") on the **Manager Profiles ScreenAssign Manager Screen**.
    - Otherwise, write an appropriate error message.
- ***Return to Main Menu*** button click: Call the Display Main Menu task.

Assign Stores

Task Decomp

Lock Types: 2 reads, 4 writes
Number of Locks: None required
Enabling Conditions: Triggered from **Main Menu Screen**
Frequency: Low
Consistency (ACID): Not critical. Not required.
Subtasks: Six subtasks. Two read-only tasks and four insert/delete tasks.



Abstract Code

(Called from the **Main Menu Screen**)

- Display the **Store Assignment Screen**.
- Upon:
  - ***View Manager Assignments*** button click:
    - Clear the output area and any success/error messages on the **Store Assignment Screen**.
    - Read the **EMAIL ADDRESS** input field on the **Store Assignment Screen**.
      - If the field is non-empty, check to see if there is a **Manager** instance with that **Manager**.*EmailAddress*.
        - If so, find all **Store** instances that are related to the **Manager** by the **Manages** relationship (using **Store**.*StoreNumber* and **Manager**.*EmailAddress*), and list the **Manager**'s **Manager**.*EmailAddress* followed by all the **Store**.*StoreNumber*s (in ascending order) in the output area of the **Store Assignment Screen**, and write a success message.
        - Otherwise, write an appropriate error message on the **Store Assignment Screen**.
      - Otherwise, write an appropriate error message on the **Store Assignment Screen**.
  - ***View Store Assignments*** button click:
    - Clear the output area and any error messages on the **Store Assignment Screen**.
    - Read the **STORE NUMBER** input field on the **Store Assignment Screen**.
      - If the field is non-empty, check to see if there is a **Store** instance with that **Store**.*StoreNumber*.
        - If so, find all **Manager** instances that are related to the **Store** by the **Manages** relationship (using **Store**.*StoreNumber* and **Manager**.*EmailAddress*), and list the **Store**'s

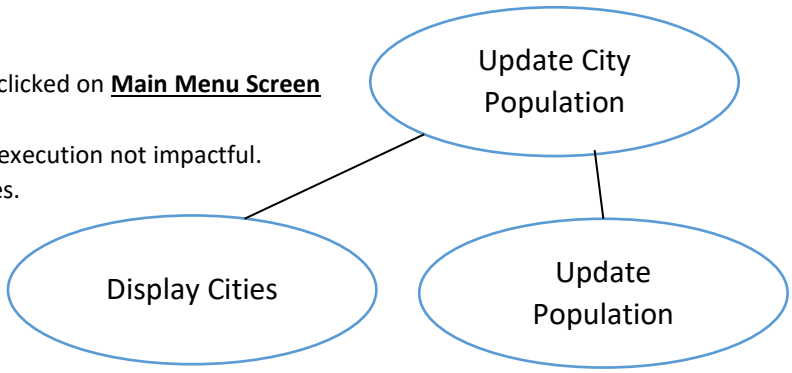
store number followed by all the **Managers**' **Manager**.*EmailAddresses* (in ascending order) in the output area of the **Store Assignment Screen**, and write a success message.
    - Otherwise, write an appropriate error message on the **Store Assignment Screen**.
  - Otherwise, write an appropriate error message on the **Store Assignment Screen**.
- ***Assign Manager to Store*** button click:
  - Clear the output area and any success/error messages on the **Store Assignment Screen**.
  - Read the **EMAIL ADDRESS** and **STORE NUMBER** input fields on the **Store Assignment Screen**.
    - If both fields are non-empty, check to see if there is a **Manager** instance with that **Manager**.*EmailAddress* and a **Store** instance with that **Store**.*StoreNumber*.
      - If so, check to see if there is already a **Manager** instance with that **Manager**.*EmailAddress* and a **Store** instance with that **Store**.*StoreNumber* that are related by the **Manages** relationship.
        - If not, create a **Manages** instance for that **Manager**.*EmailAddress* and that **Store**.*StoreNumber*, and write a success message.
        - Otherwise, write an appropriate error message on the **Store Assignment Screen**.
    - Otherwise, write an appropriate error message.
- ***Assign Store to Manager*** button click:
  - Clear the output area and any success/error messages on the **Store Assignment Screen**.
  - Read the **EMAIL ADDRESS** and **STORE NUMBER** input fields on the **Store Assignment Screen**.
    - If both fields are non-empty, check to see if there is a **Manager** instance with that **Manager**.*EmailAddress* and a **Store** instance with that **Store**.*StoreNumber*.
      - If so, check to see if there is already a **Manager** instance with that **Manager**.*EmailAddress* and a **Store** instance with that **Store**.*StoreNumber* that are related by the **Manages** relationship.
        - If not, create a **Manages** instance for that **Manager**.*EmailAddres s*and that **Store**.*StoreNumber*, and write a success message.
        - Otherwise, write an appropriate error message on the **Store Assignment Screen**.
    - Otherwise, write an appropriate error message.
- ***Unassign Manager from Store*** button click:
  - Clear the output area and any success/error messages on the **Store Assignment Screen**.
  - Read the **EMAIL ADDRESS** and **STORE NUMBER** input fields on the **Store Assignment Screen**.
    - If both fields are non-empty, check to see if there is a **Manager** instance with that **Manager**.*EmailAddress* and a **Store** instance with that **Store**.*StoreNumber*.
      - If so, check to see if that **Manager** (**Manager**.*EmailAddress*) and **Store** (**Store**.*StoreNumber*) are already related by the **Manages** relationship.
        - If so, delete the **Manages** instance for that **Manager**.*EmailAddress* and that **Store**.*StoreNumber*, and write a success message.
        - Otherwise, write an appropriate error message on the **Store Assignment Screen**.
    - Otherwise, write an appropriate error message.
- ***Unassign Store from Manager*** button click:
  - Clear the output area and any success/error messages on the **Store Assignment Screen**.
  - Read the **EMAIL ADDRESS** and **STORE NUMBER** input fields on the **Store Assignment Screen**.
    - If both fields are non-empty, check to see if there is a **Manager** instance with that **Manager**.*EmailAddress* and a **Store** instance with that **Store**.*StoreNumber*.
      - If so, check to see if that **Manager** (**Manager**.*EmailAddress*) and **Store** (**Store**.*StoreNumber*) are already related by the **Manages** relationship.
        - If so, delete the **Manages** instance for that **Manager**.*EmailAddress* and that **Store**.*StoreNumber*, and write a success message.

        - Otherwise, write an appropriate error message on the **Store Assignment Screen**.
    - Otherwise, write an appropriate error message.
- ***Return to Main Menu*** button click: Call the Display Main Menu task.

Update City Population

Task Decomp

Lock Types: Write (update) **City**
Number of Locks: None required
Enabling Conditions: ***City Population*** link clicked on **Main Menu Screen**
Frequency: Low
Consistency (ACID): Not Critical. Order of execution not impactful.
Subtasks: Two, separating reads and writes.



Abstract Code

(Called from the **Main Menu Screen**)

- Display **City Population Screen**
- For each **City** (**City**.*Location.Name*), display **City**.*Location.Name*, **City**.*Location.State*, **City**.*Population*
- User may make entries in **CITY NAME**, **STATE**, and **POPULATION** input fields.
- Upon:
  - ***Update*** button clicked:
    - Clear all success and error messages.
    - If the **CITY NAME** and **STATE** input field entries are non-empty and hold an existing **City**.*Location* combination, and if the **POPULATION** input field is non-empty and holds a valid integer value:
      - Update the **City**.*Population* value for the City instance identified by the input field values and display a success message.
    - Otherwise, display an appropriate error message.
  - ***Return to Main Menu*** button click: Call the Display Main Menu task.

Display Manufacturer Summary

Task Decomp

Lock Types: Look up **Manufacturer**.*Name*, **Product**.*PID*, and **Product**.*Price*. All three are read-only.
Number of Locks: 2 read-only locks for **Manufacturer** and **Product**.
Enabling Conditions: Successful login.
Frequency: Medium to medium-high. Will be performed more frequently than most write tasks.
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.

Display Manufacturer Summary

Abstract Code

(Called from the **Main Menu Screen**)

- Display **Manufacturers' Product Report Screen**.
- Query for information about each **Manufacturer** and their **Product**s where **Manufacturer**.*Name* is the identifier for each **Manufacturer**, **Product**.*PID* is the identifier for each **Product**, and they are related by the **ManufacturedBy** relationship. Place all following data for each **Manufacturer** in a list:
  - **Manufacturer**.*Name*
  - Total number of **Product**s offered by the **Manufacturer** (count of **Product**.*PID*s in the **ManufacturedBy** relationship with the **Manufacturer**'s **Manufacturer**.*Name*)
  - Average retail price (AVG(**Product**.*Price*)) of all the products (**Product**.*PID*) **ManufacturedBy** by the **Manufacturer** (**Manufacturer**.*Name*)
  - Minimum retail price of the manufacturer's products where **Product**.*PID* is **ManufacturedBy** Manufacturer.name)
  - Maximum retail price (MIN(**Product**.*Price*) of the manufacturer's products (as above)
- Sort the results by average retail price descending (with the highest average price appearing at the top). Display the top 100 rows on the **Manufacturers' Product Report Screen**.
  - Place a hyperlink ("***Drill-Down Link***") on each line, associating the line's text and **Manufacturer**.*Name* with the hyperlink.
- Upon:
  - ***Drill-Down Link*** click: Run the Display Manufacturer Drill-Down Details task, sending it the text and **Manufacturer**.*Name* associated with the clicked hyperlink.
  - ***Return to Main Menu*** button click: Run the Display Main Menu task.

Display Manufacturer Drill-Down Details

Task Decomp

Lock Types: Look up **Manufacturer**.*Name*, **Product**.*PID*, **Product**.*Price*, and **Category**.*Name*. All four are read-only.
Number of Locks: 3 read-only locks for **Manufacturer**, **Product**, and **Category**.
Enabling Conditions: Successful Login and clicking hyperlink for desired **Manufacturer**.*Name* on the **Manufacturers' Product Report Screen**.
Frequency: Medium. Will be performed more frequently than most write tasks, but not as frequently as **Get Manufacturer Summary** task.
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is not needed. However, the **Display Manufacturer Summary** task must be performed before any **Get Manufacturer Drill-down Details** tasks are performed.

Display Manufacturer Drill-Down Details

Abstract Code

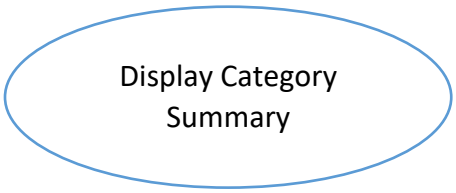
(Called from the **Manufacturers' Product Report Screen** by the Display Manufacturer Summary task)

- Display the **Manufacturer Drill-Down Details Screen**.
- Retrieve the **Manufacturer**.*Name* and text passed from the Display Manufacturer Summary task.
- Using the **Manufacturer**.*Name*, get the **Manufacturer**.*MaximumDiscount*, and display the **Manufacturer**.*Name*, and the **Manufacturer**.*MaximumDiscount* in the header of the **Manufacturer Drill-Down Details Screen**, followed by the text from passed from the Display Manufacturer Summary task.
- Find and store, for each **Product** (**Product**.*PID*) **ManufacturedBy** **Manufacturer**.*Name*, the **Product**.*PID*, **Product**.*Name*, and **Product**.*Price*, along with their **CategorizedBy** **Category**(ies) (**Category**.*Name*).
    - If there is more than one category, concatenate the category names.
- Sort the results by **Product**.*Price* descending (with the highest price appearing at the top).
- Display the product information in sorted order in the details area of the **Manufacturer Drill-Down Details Screen**.
- Upon:
    - ***Return to Manufacturers' Product Report*** button click: close the **Manufacturer Drill-Down Details Screen** and run Display Manufacturer Summary task.
    - ***Return to Main Menu*** button click: run Display Main Menu task.

Display Category Summary

Task Decomp

Lock Types: Lookup **Category**.*Name*, **Product**.*PID*, **Product**.*Price*, and **Manufacturer**.*Name*, all are read-only.
Number of Locks: No locks needed.
Enabling Conditions: Trigger by click from **Main Menu Screen**.
Frequency: Low.
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is not needed. Subtasks are not needed.



Abstract Code

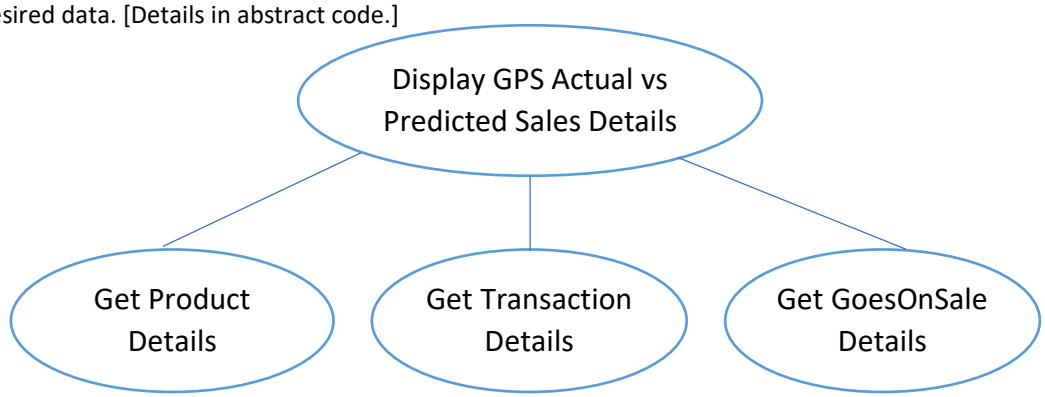
(Called from the **Main Menu Screen** and below)

- Display **Category Report Screen**
- Query database for a sorted (ascending) list of all **Category**.*Names*
  - For each **Category**.*Name* (in sorted order):
    - Get a list of **Product**s (**Product**.*PID*) **CategorizedBy** that **Category**.*Name* along with the count of those **Product**s. Display the **Category**.*Name* and the count in the areas provided on **Category Report Screen**.
      - Query the database to find the count of distinct **Manufacturer**s related to those **Product**s by the **ManufacturedBy** relationship (using **Product**.*PID* and **Manufacturer**.*Name*). Display that count in the area provided on the **Category Report Screen**.
      - Query the database to find the average **Product**.*Price* of all those **Product**s , and display that average in the area provided on the **Category Report Screen**.
- Upon ***Return to Main Menu*** button click, call the Display Main Menu task.

Display Actual vs Predicted GPS

Task Decomp

Lock Types: no locks
Number of Locks: none
Enabling Conditions: Triggered from click on ***Actual/Predicted GPS Report*** link on the **Main Menu Screen**
Frequency: Low
Consistency (ACID): not required. these are read-only tasks
Subtasks: Three sub-tasks. All read-only. First task to get product details, second and third tasks using those details to get **GoesOnSale** details. Once these details are retrieved the main task then uses in built business logic to deliver desired data. [Details in abstract code.]



Abstract Code

(Called from the **Main Menu Screen**)

- Display the **GPS Report Screen**
- For each **Product** **CategorizedBy** **Category**.*Name* = "GPS":
  - Initialize as variables:
    - $CumulativeUnitsSoldForProduct = 0
    - $MoneyFromActualUnitsSold = 0
    - $MoneyFromPredictedUnitsSold = 0
    - $TotalDiscountUnitsSold = 0
  - For each entry of **StoreSellsProduct**.*Quantity*[2] ($G) for this product
    - (via the **Product** relationship of **StoreSellsProduct**) Add **StoreSellsProduct**.*Quantity* to (running total of) $CumulativeUnitsSoldForProduct
    - Identify the **TransactionDate** ($H)
    - If the **Product** was on sale on **TransactionDate** ($H) as identified by the **GoesOnSale** relationship, then
      - Add $G to (running total of) $TotalDiscountUnitsSold
      - Add $G * **GoesOnSale**.*SalePrice* to (running total of) $MoneyFromActualUnitsSold
    - Multiply $G * **Product**.*price* * 0.75 and add to (running total of) $MoneyFromPredictedUnitsSold
  - Display (for this product):
    - **Product**.*PID*
    - **Product**.*name*
    - **Product**.*price*
    - $CumulativeUnitsSoldForProduct (derived above)

[2] Essentially, iterate through every store that has sold this product and identify the quantity sold in each.

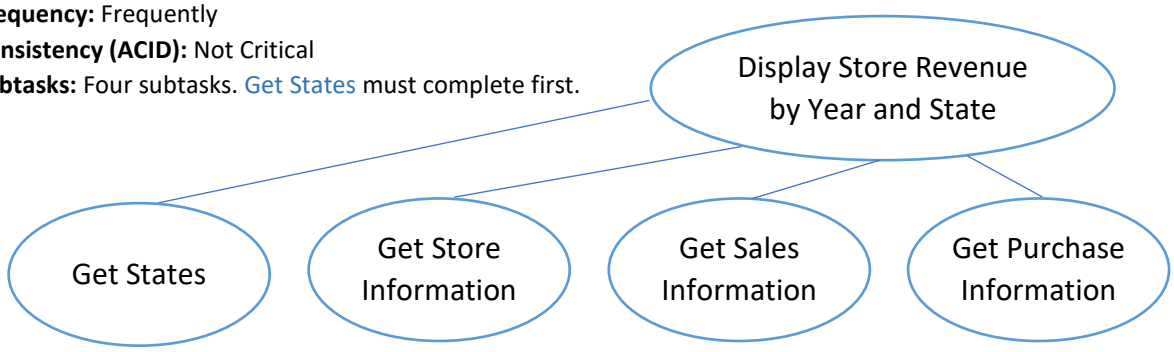
- $TotalDiscountUnitsSold (derived above)
- $MoneyFromPredictedUnitsSold (derived above)
- If $MoneyFromPredictedUnitsSold > $MoneyFromActualUnitsSold + $5000 or $MoneyFromPredictedUnitsSold < $MoneyFromActualUnitsSold - $5000
    - Display: $MoneyFromActualUnitsSold - $MoneyFromPredictedUnitsSold

- Upon a ***Return to Main Menu*** button click: Call the Display Main Menu task.

Display Store Revenue by Year and State

Task Decomp

Lock Types: None
Number of Locks: None
Enabling Conditions: ***Store Revenue Report*** link clicked on **Main Menu Screen**
Frequency: Frequently
Consistency (ACID): Not Critical
Subtasks: Four subtasks. Get States must complete first.



Abstract Code

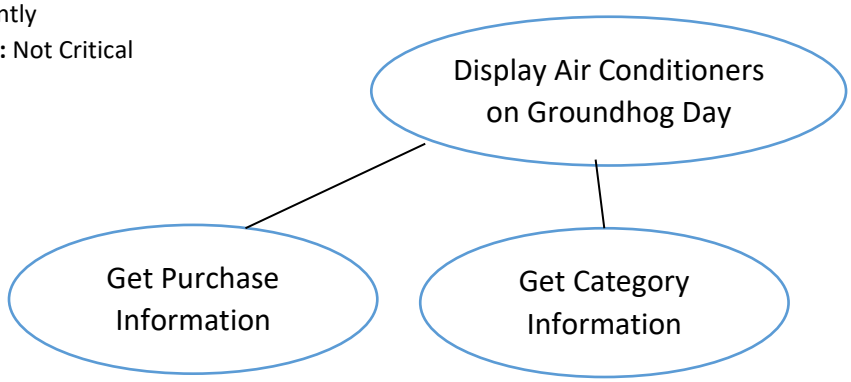
(Called from the **Main Menu Screen**)

- Display the **Store Revenue by Year and State Report Screen**
- Find distinct states from **City**.*Location*.*State* ordered ascending
- Fill state dropdown box with distinct states
- Upon:
  - ***Generate Report*** button clicked:
    - Clear the main area of the **Store Revenue by Year and State Report Screen**.
    - Read selected state from dropdown box.
    - Find distinct years (based on **StoreSellsProduct**.**TransactionDate**.*Occurrence*) when any **Store** (**Store**.*StoreNumber*) **LocatedIn** the selected state sold (**StoreSellsProduct**) any **Product** (**Product**.*PID*), ordered by year ascending.
    - For each year in order:
      - In the main area of the **Store Revenue by Year/State Report Screen**, write a blank line and then write the year to a new line.
      - For each **Store** (**Store**.*StoreNumber*) **LocatedIn** the selected state:
        - Calculate a running total of revenue ($REVENUE) by summing $AMOUNT for each **StoreSellsProduct** instance of that **Store** having **TransactionDate**.*Occurrence* in that year, where $AMOUNT is calculated as follows:
          - If the instance's **Product**.*PID* and the **Transaction Date**.*Occurrence* are related by the **GoesOnSale** relationship:
            - $AMOUNT = **StoreSellsProduct**.*Quantity* * **GoesOnSale**.*SalePrice*
          - Else:
            - $AMOUNT = **StoreSellsProduct**.*Quantity* * **Product**.*Price*
        - Store the **Store** along with its total $REVENUE
      - Sort the list of **Store**s and $REVENUE by $REVENUE descending.
      - Write the sorted list of **Store**s and $REVENUE under the year in the main area of the **Store Revenue by Year and State Report Screen**
  - ***Return to Main Menu*** button click, call the Display Main Menu task.

Display Air Conditioners on Groundhog Day

Task Decomp

Lock Types: None
Number of Locks: None
Enabling Conditions: ***GH Day AC Report*** link clicked on **Main Menu Screen**
Frequency: Frequently
Consistency (ACID): Not Critical
Subtasks: Two



Abstract Code

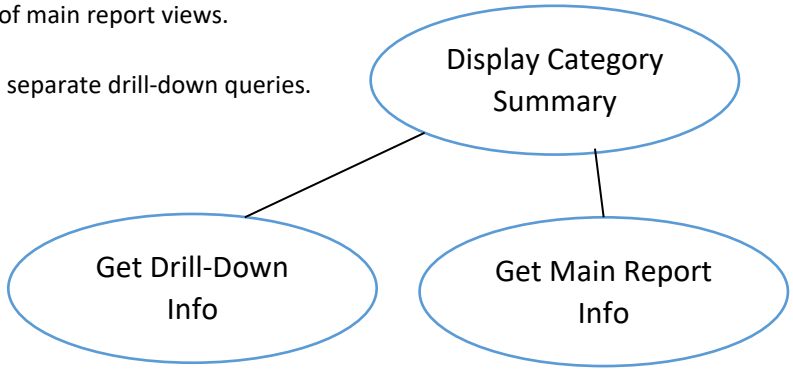
(Called from the **Main Menu Screen**)

- Display the **Air Conditioners on Groundhog Day Report Screen**
- Upon:
  - ***Generate Report*** button clicked:
    - Clear the main area of the **Air Conditioners on Groundhog Day Report Screen**.
    - Find distinct years (based on **StoreSellsProduct**.**TransactionDate**.*Occurrence*) when any **Store** (**Store**.*StoreNumber*) sold (**StoreSellsProduct**) any **Product** (**Product**.*PID*) **CategorizedBy** **Category**.*Name* = "Air Conditioning", ordered by year ascending.
    - For each year in order:
      - Get the total quantity ($TOTAL_QUANTITY = SUM(**StoreSellsProduct**.*Quantity*)) of **Product**s (**Product**.*PID*) **CategorizedBy** **Category**.*Name* = "Air Conditioning" during that year (based on **StoreSellsProduct**.**TransactionDate**.*Occurrence*).
      - Set $AVERAGE_PER_DAY = . $TOTAL_QUANTITY / 365.0.
      - Get the quantity of air conditioning products sold on 2/2 of that year ($GH_QUANTITY = SUM(**StoreSellsProduct**.*Quantity*)) of **Product**s (**Product**.*PID*) **CategorizedBy** **Category**.*Name* = "Air Conditioning" on Groundhog Day during that year (based on **StoreSellsProduct**.**TransactionDate**.*Occurrence* month and day is February 2).
      - On the **Air Conditioners on Groundhog Day Report Screen**, display the year, $AVERAGE_PER_DAY, and $GH_QUANTITY, separating the years' data with blank lines.
  - ***Return to Main Menu*** button click, call the Display Main Menu task.

Display State Volume by Category

Task Decomp

Lock Types: Lookup **TransactionDate**.*Occurrence*, **Product**.*PID*, **Category**.*Name*, **Store**.*StoreNumber*, **Store**.*StreetAddress*, **City**.*Location.Name*, **Manager**.*EmailAddress*, **Manager**.*Name*, and **Manufacturer**.*Name*, all are read-only.
Number of Locks: No locks needed.
Enabling Conditions: Trigger by click from **Main Menu Screen**. Drill-down dependent on display of main report.
Frequency: Low. Number of drill-downs independent of main report views.
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is not required. Use subtask to separate drill-down queries.



Abstract Code

(Called from the **Main Menu Screen** and below)

- Display **State Volume by Category Report Screen**.
- Query the database for distinct month/years represented in the **StoreSellsProduct** relationship (based on **TransactionDate**.*Occurrence*), ordered by year, month descending (not required, but will put latest date on top), and use this to populate the month/year dropdown list in the **State Volume by Category Report Screen**. Select the first month/year in the dropdown list.
- User will select a month/year from the dropdown list.
- Upon:
  - ***Generate Report*** button click:
    - Clear any previously written lines.
    - Read the selected month/year from the dropdown list in the **State Volume by Category Report Screen**.
    - Query database for a sorted (ascending) list of all **Category**.*Names*.
      - For each **Category**.*Name* (in sorted order):
        - Query the database to find all states (**City**.*Location.State*) related by **LocatedIn** to the **StoreSellsProduct** instances' stores (**StoreSellsProduct**.**Store**.*StoreNumber*) and the total number of units sold (SUM(**StoreSellsProduct**.*Quantity*)), for **Product**s (**StoreSellsProduct**.**Product**.*PID*) **CategorizedBy** the given **Category**.*Name* and sold in the selected month/year (based on **StoreSellsProduct**.**TransactionDate**.*Occurrence*), grouped by state and ordered by units sold.
        - For each state having the highest number of units sold, display the **Category**.*Name*, the **City**.*Location.State*, and the number of units sold as a new line in the area provided on the **State Volume by Category Report Screen**, along with a button or link ("***Drill-Down Button***") associated with that line's category and state.
          - If there are no units sold of that category during that month/year, display the **Category**.*Name* and "No units sold."
    - Upon click of any of the ***Drill-Down Button***s:
      - Clear the drill-down area provided at the bottom of the **State Volume by Category Report Screen**.
      - Determine the category and state associated with the clicked button.

- Read the selected month/year from the dropdown list.
- Query the database to find all **Store**s (**Store**.*StoreNumber*) that participated in the **StoreSellsProduct** relationship on **Date**s (**TransactionDate**.*Occurrence*) during the selected month/year, for **Product**s (**Product**.*PID*) **CategorizedBy** the associated **Category**.*Name*.
  - For each such **Store**:
    - Query the database using the **Store**.*StoreNumber* to find the **Store**'s **Store**.*StreetAddress*, and **City**.*Location.Name* (from **LocatedIn** relationship).
    - Write a new line in the drill-down area provided at the bottom of the **State Volume by Category Report Screen** that includes the **Store**.*StoreNumber*, **Store**.*StreetAddress*, and **City**.*Location.Name*.
    - Using the **Store**.*StoreNumber* and the **Manages** relationship, find all **Manager**.*EmailAddress*es and **Manager**.*names* of **Manager**s who manage that store. Append these **Manager**s' names and e-mail addresses to the store information in the **State Volume by Category Report Screen** (one manager per line if more than one).

- ***Return to Main Menu*** button click, call the Display Main Menu task.

Display Annual Average Revenue by City Population

Task Decomp

Lock Types: Look up **City**.*Location*, **City**.*Population*, **TransactionDate**.*Occurrence*, **Product**.*PID*, **Product**.*Price*, and **StoreSellsProduct**.*Quantity*, all read-only.
Number of Locks: Read-only locks for **City**, **TransactionDate**, **Product**, and **StoreSellsProduct**.*Quantity*.
Enabling Conditions: Successful Login
Frequency: Low
Consistency (ACID): Not critical, order is not critical. The task picks up the current values of population for each city. There is no population change tracking requirement.
Subtasks: Mother Task is not needed. No decomposition needed.

Display Annual Average Revenue by City Population

Abstract Code

(Called from the **Main Menu Screen**)

- Display the **Annual Average Revenue by City Population Report Screen**.
- For each of four population categories ("Small" < 3,700,000, "Medium" >= 3,700,000 and < 6,700,000, "Large" >= 6,700,000 and < 9,700,000, and "Extra Large" >= 9,700,000), and each year (based on **StoreSellsProduct**.**TransactionDate**.*Occurrence*) for all **StoreSellsProduct** instances, calculate the $AVERAGE ANNUAL REVENUE (total sales during a given year for all stores located in cities within the population category, divided by the number of cities with stores in that population category), as further described below.
  - To calculate total sales for a **Store** (**Store**.*StoreNumber*) within a given year, look at each **StoreSellsProduct** instance for that **Store** and find the related **Product**.*PID* and **TransactionDate**.*Occurrence*.
    - Exclude the sale If the **TransactionDate**.*Occurrence* is not within the given year.
    - If the **Product**.*PID* and **TransactionDate**.*Occurrence* are related by **GoesOnSale**:
      - Define $TRANSACTIONAMOUNT = **GoesOnSale**.*SalePrice* * **StoreSellsProduct**.*Quantity*
    - Otherwise:
      - Define $TRANSACTIONAMOUNT = **Product**.*Price* * **StoreSellsProduct**.*Quantity*
    - Check whether the **Store**.*StoreNumber* is **LocatedIn** a **City** whose **City**.*Population* is within the specified range.
      - If so, count the sale for that year and population category.
      - If not, exclude the sale for that year and population category.
  - For each year and population category, calculate the sum of all $TRANSACTIONAMOUNTS for each store in that population category during that year.
  - To determine the number of cities within each population category for each year, count the distinct cities (**City**.*Name*) where **Store**s are **LocateIn** that have **StoreSellsProduct** instances with **TransactinDate**.*Occurrence* within that given year
  - Sort the distinct years represented in any of the **StoreSellsProduct** instances in descending order and display these has headings for columns of a grid. Display the row headings of the grid as "Small", "Medium", "Large", and "Extra Large". Display the respective calculated $AVERAGE ANNUAL REVENUE in each cell.
- Upon:
  - ***Return to Main Menu*** button click: run Display Main Menu task.