Table of Contents:

Data Types:

STORE

Attribute	**Data type**	**Nullable**
Store_Number	String	Not Null
Phone_Number	String	Not Null
Street_Address	String	Not Null
Has_Restaurant	Boolean	Not Null
Has_Snack_Bar	Boolean	Not Null

CHILDCARE

Attribute	**Data type**	**Nullable**
Limit	Integer	Not Null

CITY

Attribute	**Data type**	**Nullable**
City_Name	String	Not Null
State_Location	String	Not Null
Population	Integer	Not Null
Population_Size_Category	String	Not Null

PRODUCT

Attribute	**Data type**	**Nullable**
PID	String	Not Null
Product_Name	String	Not Null
Retail_Price	Float	Not Null

CATEGORY

Attribute	**Data type**	**Nullable**
Category_Name	String	Not Null

DAY

Attribute	**Data type**	**Nullable**
Date	String	Not Null

DISCOUNT

Attribute	**Data type**	**Nullable**
Discount_Price	Float	Not Null

SALE

Attribute	**Data type**	**Nullable**
Quantity	Integer	Not Null
Total_Amount	Float	Not Null

HOLIDAY

Attribute	**Data type**	**Nullable**
Name	List<String>	Not Null

ADVERTISING_CAMPAIGN

Attribute	**Data type**	**Nullable**
Description	String	Not Null

Business Logic Constraints:

- No user authentication is needed.

STORE

- Stores offering food should have a restaurant, a snack bar, or both.
- At most of LEOFURN's stores, the restaurant is located before customers enter the main sales floor. Recent psychometric studies show that customers with a full stomach are less interested in purchasing certain kinds of items (such as dining room furniture, kitchen utensils, etc.) but show stronger interest in others (such as beds, couches, sofas, or reclining chairs). This behavior does not apply to snack bars because those are near the exit of the store.

CHILDCARE

- The limit is chosen by each store from predetermined values.
- All limits may need to be updated if it changes (such as from 45 minutes to 60) or a new limit requires manual updating outside of a data load.
- Offering more childcare may allow parents to spend more time browsing and purchase more items, while it's also possible that the cost of providing childcare is not offset by the additional sales it may generate and should be discontinued.
- For "Report 7 – Childcare Sales Volume", it's based on the last 12 months' worth of available sales in the system. Stores that do not offer childcare should be included on this report and grouped as "No childcare".

CITY

- A city's population can be updated by users after data for it has been loaded.
- The categories for city size are: Small (population <3,700,000), Medium (population >=3,700,000 and <6,700,000), Large (population >=6,700,000 and <9,000,000) and Extra Large (population >=9,000,000).

PRODUCT

- All products are available and sold at all stores.
- The retail price is in effect unless there is a discount price.
- If a product is discounted, it is for the same price in all stores—i.e., stores are not allowed to discount items independently or have store-specific discount prices.

CATEGORY

- “Couches and Sofas” should be one of the categories.
- “Outdoor furniture” should be one of the categories.
- “Report 1 – Category Report” should query for all categories including those without products.
- For “Report 5 – State with Highest Volume for each Category”, each category will only be listed once unless two or more states tied for selling the highest number of units in that category. Products which do not have a category may be excluded from this report.
- “Report 8 – Restaurant Impact on Category Sales” may exclude any categories that are not assigned products as their information would not be useful here.

DAY

- Assume a year is exactly 365 days.
- Outdoor furniture sales appear to spike on Groundhog Day (which falls on February 2 each year). This is probably because customers begin thinking about the warm spring weather ahead. This report does not imply storing Groundhog Day as a holiday, but to explicitly query for February 2.)
- Day cannot include dates greater than today’s date and dates before year 1900.
- If the report requires filter input from user, the system needs to generate the report based on the filter conditions.

DISCOUNT

- Discounts could occur on dates where there are no sales.
- Product discounts introduce on average a 25% increase in volume (quantity sold).
- We assume that if an item that was offered at a discount were instead offered at the retail price, the quantity of items sold would be reduced by 25%.

SALE

- For reporting purposes, sales tax values are ignored.
- The system is not required to store which products were purchased together during a single sales transaction.
- Sales could occur on dates where there are no campaigns.
- The revenue calculation takes into account the items that were sold at a discount.

- For “Report 2 – Actual versus Predicted Revenue for Couches and Sofas”, only predicted revenue differences greater than $5000 (positive or negative) should be displayed.

HOLIDAY

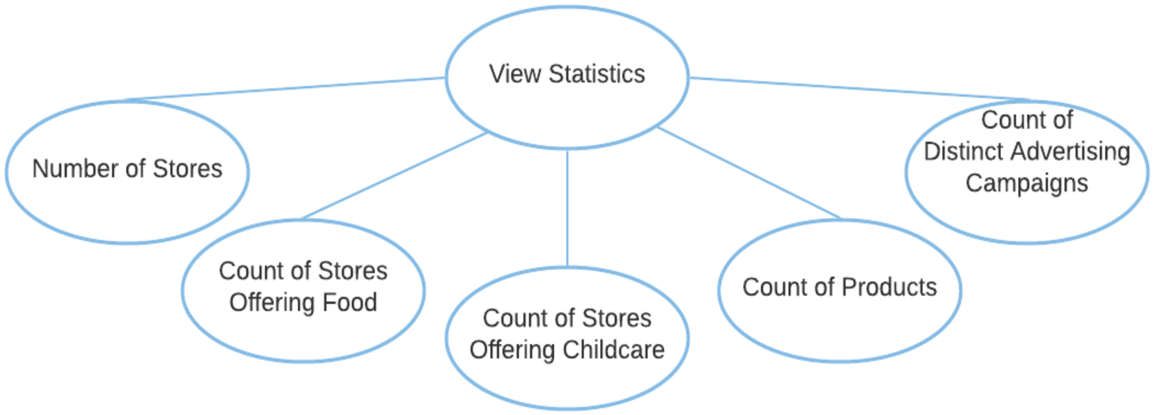
- Holiday information can be added by users after data for it has been loaded.
- User can view and add holiday information but can’t delete holiday information.

ADVERTISING_CAMPAIGN

- While there are numerous possibilities to correlate advertising campaigns and sales data, the LEOFURN team would like to start out with a relatively simple analysis: if a product is discounted, does an advertising campaign affect its sales volume?
- For “Report 9 – Advertising Campaign Analysis”, all products sold at a discount price are included. The final report output only displays the top 10 and followed by the bottom 10 results when sorting the results by difference in descending order.

View Statistics

Task Decomp


View Statistics
Number of Stores
Count of Stores Offering Food
Count of Stores Offering Childcare
Count of Products
Count of Distinct Advertising Campaigns

Lock Types: 4 Read-only locks on STORE, CHILDCARE, PRODUCT, and ADVERTISING_CAMPAIGNS tables.
Number of Locks: Several different schema constructs are needed.
Enabling Conditions: None – all five statistics are available to show once the page is loaded and menu is displayed.
Frequency: High – all five statistics are shown every time when the main page is loaded.
Consistency (ACID): Not critical, order is not critical.
Subtasks: All tasks must be done but can be done in parallel. Mother task is required to coordinate subtasks. Order is not necessary.

Abstract Code

- Show *“**Holiday Maintenance**”, “**View Product by Category Report**”, “**View Actual vs. Predicted Revenue for Couches and Sofas Report**”, “**View Store Revenue by Year by State Report**”, “**View Groundhog Day Outdoor Furniture Report**”, “**View State with Highest Volume by Category Report**”, “**View Revenue by Population Report**”, “**View Childcare Sales Volume Report**”, “**View Restaurant Impact on Category Sales Report**”, “**View Advertising Campaign Analysis Report**”,* and *“**Population Maintenance**”* buttons/links on the **Dashboard** form.
- Upon:
    - Click ***Holiday Maintenance*** button – Jump to the **Holiday Maintenance** form.

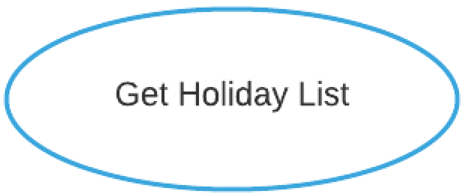
- Click ***View Product by Category Report*** button – Jump to the **View Product by Category Report** task.
- Click ***View Actual vs. Predicted Revenue for Couches and Sofas Report*** button – Jump to the **View Actual vs. Predicted Revenue for Couches and Sofas Report** task.
- Click ***View Store Revenue by Year by State Report*** button – Jump to the **Get Available State List** task.
- Click ***View Groundhog Day Outdoor Furniture Report*** button – Jump to the **View Groundhog Day Outdoor Furniture Report** task.
- Click ***View State with Highest Volume by Category Report*** button – Jump to the **Get Year and Month List** task.
- Click ***View Revenue by Population Report*** button – Jump to the **View Revenue by Population Report** task.
- Click ***View Childcare Sales Volume Report*** button – Jump to the **View Childcare Sales Volume Report** task.
- Click ***View Restaurant Impact on Category Sales Report*** button – Jump to the **View Restaurant Impact on Category Sales Report** task.
- Click ***View Advertising Campaign Analysis Report*** button – Jump to the **View Advertising Campaign Analysis Report** task.
- Click ***Population Maintenance*** button – Jump to the **Population Maintenance** form.

- Display statistics for “the count of stores”, “count of stores offering food (have a restaurant, a snack bar, or both)”, “count of stores offering childcare”, “count of products”, and “count of distinct advertising campaigns” on the **Dashboard** form.
    - Show “the count of stores”.
        - Query for total count of Store_Number in the STORE table.
        - Display the total count.
    - Show “count of stores offering food”.
        - Query for the total count of Store_Number in the STORE table that has either or both Has_Restaurant and Has_Snack_Bar value as true.
        - Display the total count.
    - Show “count of stores offering childcare”.
        - Query for the total count of Store_Number in the STORE table that has a Childcare center association in the CHILDCARE table.
        - Display the total count.

- Show “count of products”.
  - Query for the total count of PID in the PRODUCT table.
  - Display the total count.
- Show “count of distinct advertising campaigns”.
  - Query for the total count of Description in ADVERTISING_CAMPAIGNS table.
  - Display the total count.

Get Holiday List

Task Decomp



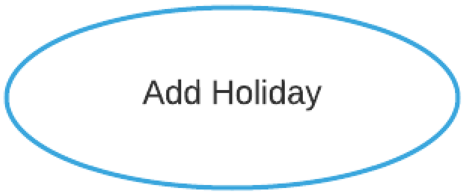
Lock Types: Read-only on HOLIDAY table.
Number of Locks: Single.
Enabling Conditions: Triggered when ***Holiday Maintenance*** button is clicked.
Frequency: Low.
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code

- User clicked on the ***Holiday Maintenance*** button from the **Dashboard** form.
- Run the **Get Holiday List** task: query for information about the available Name field from the HOLIDAY table.
  - Display the holiday name list.
- Upon:
  - User enters Holiday Name (‘$HolidayName’) in input textbox and select Date (‘$HolidayDate’) in Calendar Dropdown.
  - Click ***Add Holiday*** button –
    - Jump to the **Add Holiday** task.
- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

Add Holiday

Task Decomp



Lock Types: 2 write-only locks on HOLIDAY and DAY tables.
Number of Locks: Several different schema constructs are needed.
Enabling Conditions: Triggered when ***Add Holiday*** button is clicked on **Holiday Maintenance** form.
Frequency: Low.
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.

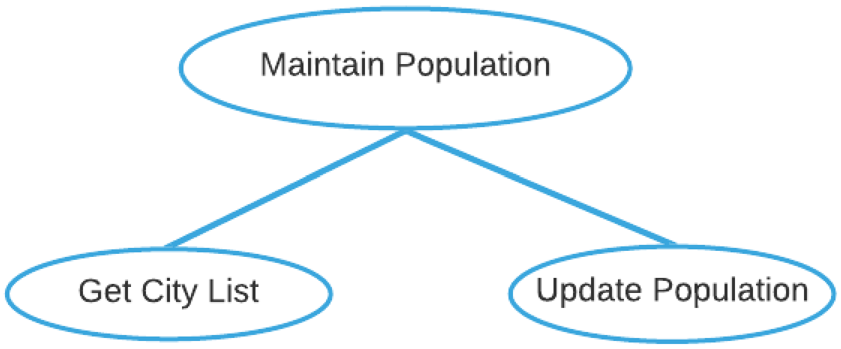
Abstract Code

- User enters Holiday Name ('$HolidayName') in input textbox and select Date ('$HolidayDate') in Calendar Dropdown.
- Click ***Add Holiday*** button.
- Run the **Add Holiday** task:
    - If data validation passed for both holiday name and date in Client Side, then:
        - If same holiday name and same holiday date exist:
            - Go back to **Holiday Maintenance** form and show the failure message this holiday with this date existed.
        - If Holiday Name is not existed but date exists:
            - Store the Holiday Name in HOLIDAY table and link its date with DAY table.
            - Go back to **Holiday Maintenance** form and show success message.
        - If both Holiday Name and date do not exist:
            - Store the *Holiday Name* and *date* in both HOLIDAY and DAY tables.
            - Go back to **Holiday Maintenance** form and show success message.

- Else: display the invalid error message in **Holiday Maintenance** form.
- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

Maintain Population

Task Decomp



Lock Types: Read and write on CITY table.
Number of Locks: Two different schema constructs are needed.
Enabling Conditions: Triggered when ***Population Maintenance*** button is clicked.
Frequency: Low – Both two have the same frequency.
Consistency (ACID): Critical. If the city population is updated by the user, population category in Revenue by Population Report needs to be updated to make the database consistent.
Subtasks: Mother Task is required to coordinate subtasks. Order is necessary. **Get City List** first followed by **Update Population**.

Abstract Code

- User clicked on the ***Population Maintenance*** button from the **Dashboard** form.
- Run the **Get City List** task: query for information about the available *State_Location*, *City_Name,* and *Population* fields from the CITY table.
  - Display *State_Location* list in ascending order in the drop-down list.
  - Select state and then display the available *City_Name* (from the CITY table) list in the drop-down list.
  - Select target *City_Name*, and display *Population* in view population textbox.
- User edited population textbox.

- Upon:
    - Click ***Update Population*** button –
        - Jump to the **Update Population** task.
- Run the **Update Population** task.
    - If data validation passed for *Population* in Client Side, then:
        - If the updated population entered is the same as the original population, do nothing.
        - Else if the updated population entered is different from the original population, update the *Population* and *Population_Size_Category* in the CITY table.
        - Else updated population is not entered, populate a message asking for user input.
    - Else: display the invalid error message in **Population Maintenance** form.
- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

View Product by Category Report

Task Decomp

Lock Types: 2 Read-only locks on CATEGORY and PRODUCT tables.
Number of Locks: Several different schema constructs are needed.
Enabling Conditions: Triggered when ***View Product by Category Report*** button is clicked.
Frequency: Low.
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.

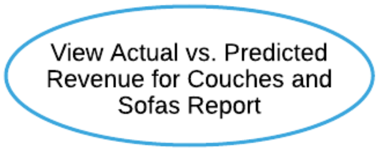
Abstract Code

- User clicked on the ***View Product by Category Report*** button from the **Dashboard** form.

- Run the **View Product by Category Report** task: query for each category from CATEGORY and PRODUCT tables, including those without products.
    - Get all Category_Name data (from the CATEGORY table).
    - For each category including those without products:
        - Find minimum, average, and maximum Retail_Price data for all products (from the PRODUCT table).
        - Find total number of products by counting their PID data (from the PRODUCT table).
    - Sort by category name in ascending order.
- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

View Actual vs. Predicted Revenue for Couches and Sofas Report

Task Decomp



Lock Types: 5 Read-only locks on CATEGORY, PRODUCT, SALE, DAY, and DISCOUNT tables.
Number of Locks: Several different schema constructs are needed.
Enabling Conditions: Triggered when ***View Actual vs. Predicted Revenue for Couches and Sofas Report*** button is clicked.
Frequency: Low.
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code

- User clicked on the ***View Actual vs. Predicted Revenue for Couches and Sofas Report*** button from the **Dashboard** form.
- Run the **View Actual vs. Predicted Revenue for Couches and Sofas Report** task:
    - Get the category of couches and sofas (from the CATEGORY table).
    - For each product of Couches and Sofas category (from the PRODUCT table):

        - Get PID, Product_Name, and Retail_Price data
        - Get Quantity data for number of products sold (from the SALE table) at the specific Date (from the DAY table); Find total number of products ever sold by aggregating Quantity in all sale days.
        - Get Quantity data for number of products sold (from the SALE table) at the Date has a Discount_Price (from DAY and DISCOUNT tables); Find total number of products sold at a discount by aggregating Quantity in all discount dates.
        - Find total number of products sold at retail price by subtracting total number of products sold at a discount from total number of products ever sold.
        - Get actual revenue in one day using Total_Amount by multiplying Quantity and Discount_Price (from SALE and DISCOUNT tables) at the specific Date (from the DAY table); Find total actual revenue by aggregating actual revenue in all sale dates.
        - Find predicted revenue in one day by multiplying 75% Quantity and Retail_Price (from SALE and PRODUCT tables) at the specific Date (from the DAY table); Find total predicted revenue by aggregating actual revenue in all sale dates.
        - Find revenue difference by subtracting predicted revenue from actual revenue.
    - If revenue difference is greater than $5000 (positive or negative): Display revenue difference and sort in descending order.
- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

Get Available State List

Task Decomp

Lock Types: Read-only on CITY table.
Number of Locks: Single.

Enabling Conditions: Triggered when ***View Store Revenue by Year by State Report*** button is clicked.
Frequency: Low.
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code

- User clicked on ***View Store Revenue by Year by State Report*** button from the **Dashboard** form.
- Run the **Get Available State List** task: query for information about the available State_Location fields from the CITY table.
    - Display *State_Location* list in ascending order on the drop-down list.
- On the drop-down list, show ***Run Report*** button.
- Upon:
    - Click ***Run Report*** button –
        - If *state* is selected – Jump to the **View State Revenue by Year by State Report** task.
        - If *state* field is empty – Display a message asking for user input.
- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

View Store Revenue by Year by State Report

Task Decomp

Lock Types: 6 Read-only locks on CITY, STORE, SALE, PRODUCT, DAY, and DISCOUNT tables.
Number of Locks: Several different schema constructs are needed.
Enabling Conditions: Triggered when ***Run Report*** button from the drop-down list is clicked.
Frequency: Low.
Consistency (ACID): Not critical, order is not critical.

Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code

- User clicked on the ***Run Report*** button from the drop-down list.
- If data validation is successful for *state* input fields, then proceed.
- Run the **View State Revenue by Year by State Report** task:
    - Find all stores from the STORE table based on the *state* (from the CITY table) selected.
    - On every sale *date* (from the DAY table), find each product's sale revenue based on Total_Amount (from the SALE table).
        - Total_Amount (from the SALE table) is calculated based on the Date purchased (from the DAY table), the Quantity (from the SALE table), and individual item price.
        - Individual item prices can be determined by Retail_Price (from the PRODUCT table) or Discount_Price (from the DISCOUNT table) when the product has a discount.
    - In each year, at each store in the selected state, find the total revenue by aggregating all products' sale revenue on every sale date in that year.
    - Sort by sale year ascending, then sort by total revenue in descending order for each store in the selected state, display Store_Number (from the STORE table), Street_Address (from the STORE table) of the store, and City_Name (from the CITY table).
- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

View Groundhog Day Outdoor Furniture Report

Task Decomp

Lock Types: 3 Read-only locks on DAY, CATEGORY and SALE tables.
Number of Locks: Several different schema constructs are needed.
Enabling Conditions: Triggered when ***View Groundhog Day Outdoor Furniture Report*** button is clicked.
Frequency: Low.

Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code

- User clicked on the ***View Groundhog Day Outdoor Furniture Report*** button from the **Dashboard** form.
- Run the **View Groundhog Day Outdoor Furniture Report** task:
  - Get and return the year (from the DAY table).
  - For each year:
    - Get the outdoor furniture category (from the CATEGORY table).
    - Get Quantity data for number of products sold (from the SALE table) at the specific Date that year (from the DAY table); Find total number of products sold by aggregating Quantity in all sale days that year.
    - Find average number of products sold per day by dividing total number of products sold by 365.
    - Find total number of products sold on Groundhog Day (Feb 2) using Quantity and Date (from SALE and DAY tables).
  - Sort by Year in ascending order.
- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

Get Year and Month List

Task Decomp

Lock Types: Read-only on DAY table.
Number of Locks: Single.
Enabling Conditions: Triggered when ***View State with Highest Volume by Category Report*** button is clicked.
Frequency: Medium – monthly report.
Consistency (ACID): Not critical, order is not critical.

Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code

- User clicked on the ***View State with Highest Volume by Category Report*** button from the **Dashboard** form.
- Run the **Get Year and Month List** task: query for information about the available *year* and *month* fields from the DAY table.
    - Display both *year* and *month* lists in descending order on the drop-down list.
- On the drop-down list, show ***Run Report*** button.
- Upon:
    - Click ***Run Report*** button –
        - If both *year* and *month* are selected – Jump to the **View State with Highest Volume by Category Report** task.
        - If one or both fields are empty – Display a message asking for user input.
- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

View State with Highest Volume by Category Report

Task Decomp

Lock Types: 5 Read-only locks on CITY, DAY, SALE, CATEGORY, and STORE tables.
Number of Locks: Several different schema constructs are needed.
Enabling Conditions: Triggered when ***Run Report*** button from the drop-down list is clicked.
Frequency: Medium – monthly report.
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code

- User clicked on the ***Run Report*** button from the drop-down list.
- If data validation is successful for both *year* and *month* input fields, then proceed.
- Run the **View State with Highest Volume by Category Report** task:
    - Find all Sales data from the SALE table based on the *year* and *month* (from the DAY table) selected.
    - Get all categories by using Category_Name data (from the CATEGORY table).
    - In each category, aggregate each state's sales Quantity data (from the SALE table) by adding up all stores (from the STORE table)'s sales Quantity in each state (from the CITY table).
    - Find the states that sold the highest number of units in each category.
        - If two or more states tied for having the greatest number of units, then save all those states and their total units.
    - Sort by category name ascending, display each category name, its corresponding states with the highest units sold and total units sold.
- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

View Revenue by Population Report

Task Decomp

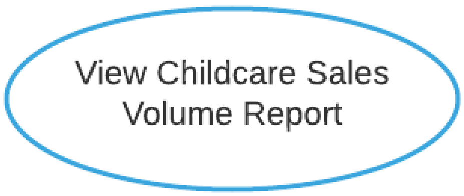
Lock Types: 4 Read-only locks on DAY, CITY, STORE, and SALE tables.
Number of Locks: Several different schema constructs are needed.
Enabling Conditions: Triggered when ***View Revenue by Population Report*** button is clicked.
Frequency: Low.
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code

- User clicked on the ***View Revenue by Population Report*** button from the **Dashboard** form.
- Run the **View Revenue by Population Report** task:
    - Find the year, Population_Size_Category and sale's Total_Amount from the DAY, SALE, STORE, and CITY tables.
    - Group by year and Population_Size_Category, and then aggregate the Total_Amount as total revenue.
    - Display Total Revenue in a tabular form:
        - Row: year, sorted in ascending order.
        - Column: Population_Size_Category, sorted in ascending order.
- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

View Childcare Sales Volume Report

Task Decomp



Lock Types: 4 Read-only locks on DAY, STORE, SALE, and CHILDCARE tables.
Number of Locks: Several different schema constructs are needed.
Enabling Conditions: Triggered when ***View Childcare Sales Volume Report*** button is clicked.
Frequency: Medium – monthly report.
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code

- User clicked on the ***View Childcare Sales Volume Report*** button from the **Dashboard** form.
- Run the **View Childcare Sales Volume Report** task:

- Find and aggregate the store monthly sales from the SALE and STORE table based on the Store_Number (from the STORE table) and month (date from the DAY and SALE tables) based on the last 12 months.
- Find the store with (or without) childcare category from the STORE and CHILDCARE tables.
- Group the total sales by month and by childcare category (no childcare offer will be grouped as category "No childcare").
- Display Total Sale in a tabular form:
    - Row: month, sorted in ascending order.
    - Column: childcare category.

- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

View Restaurant Impact on Category Sales Report

Task Decomp

Lock Types: 4 Read-only locks on CATEGORY, PRODUCT, STORE, and SALE tables.
Number of Locks: Several different schema constructs are needed.
Enabling Conditions: Triggered when ***View Restaurant Impact on Category Sales Report*** button is clicked.
Frequency: Low.
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code

- User clicked on the ***View Restaurant Impact on Category Sales Report*** button from the **Dashboard** form.
- Run the **View Restaurant Impact on Category Sales Report** task:
    - Get all Category_Name data that has product(s) assigned (from CATEGORY and PRODUCT tables).

- For each category:
  - Get products in this category (from CATEGORY and PRODUCT tables).
  - Find stores with Has_Restaurant (from the STORE table) = TRUE:
    - Display store type with value “Restaurant”.
    - Find total quantity sold by aggregating all Quantity (from the SALE table) of all products for these stores.
  - Find stores with Has_Restaurant (from the STORE table) = FALSE:
    - Display the store type with value “Non-Restaurant”.
    - Find total quantity sold by aggregating all Quantity (from the SALE table) of all products for these stores.
- Group by Category_Name ascendingly and with “Non-Restaurant” store data listed first.

- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

View Advertising Campaign Analysis Report

Task Decomp

Lock Types: 5 Read-only locks on DISCOUNT, PRODUCT, SALE, DAY, and ADVERTISING_CAMPAIGN tables.
Number of Locks: Several different schema constructs are needed.
Enabling Conditions: Triggered when ***View Advertising Campaign Analysis Report*** button is clicked.
Frequency: Low.
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code

- User clicked on the ***View Advertising Campaign Analysis Report*** button from the **Dashboard** form.

- Run the **View Advertising Campaign Analysis Report** task: query for information about quantity sold during and outside campaign for all products from DISCOUNT, PRODUCT, SALE, DAY, and ADVERTISING_CAMPAIGN tables.
    - Get Discount_Price data for all products (from the DISCOUNT table).
    - While a Product has Discount_Price:
        - Get PID, Product_Name data (from the PRODUCT table).
        - Get Quantity data for this product sold during campaign (from SALE and ADVERTISING_CAMPAIGN tables) at the specific discount Date (from DAY and DSICOUNT table); Find total quantity sold during campaign by aggregating Quantity in all discount sale days that hold a campaign.
        - Get Quantity data for this product sold outside campaign (from SALE and ADVERTISING_CAMPAIGN tables) at the specific discount Date (from DAY and DSICOUNT table); Find total quantity sold outside campaign by aggregating Quantity in all discount sale days without a campaign.
        - Find difference by subtracting quantity sold outside campaign from quantity sold during campaign.
    - Sort by difference in descending order and only display the top 10 followed by the bottom 10.
- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.