Table of Contents:

Data Types

store

Attribute	Data type	Nullable
childcareLimit	Integer	Not Null
hasRestaurant	Boolean	Not Null
hasSnackBar	Boolean	Not Null
storeID	Integer	Not Null
phoneNumber	Integer	Not Null
streetAddress	Integer	Not Null

city

Attribute	Data type	Nullable
cityID	Integer	Not Null
name	String	Not Null
state	String	Not Null
population	Integer	Not Null

product

Attribute	Data type	Nullable
PID	Integer	Not Null
name	String	Not Null
retailPrice	Float	Not Null

date

Attribute	Data type	Nullable
year	String	Not Null
month	String	Not Null
day	String	Not Null

category

Attribute	Data type	Nullable
categoryName	String	Not Null

discount

Attribute	Data type	Nullable
discountPrice	Float	Not Null
discountRate	Float	Not Null

holiday

Attribute	Data type	Nullable
holidayName	String	Not Null

advertisingCampaign

Attribute	Data type	Nullable
description	String	Not Null

Business Logic Constraints:

- A city may have multiple stores.
- Assume that all products are available and sold at all stores.
- A product must be assigned to one or more categories. A category may or may not have products associated with it.
- If a product is discounted, it is for the same price in all stores.
- If a day has multiple holidays, their names can be combined.
- The retail price is in effect unless there is a discount price.
- Campaign dates may overlap.

Main Menu

Task Decomp

Lock Types: Lookup store, product, advertisingCampaign, all are Read-only.
Number of Locks: 5
Enabling Conditions: None
Frequency: High
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.

Abstract Code

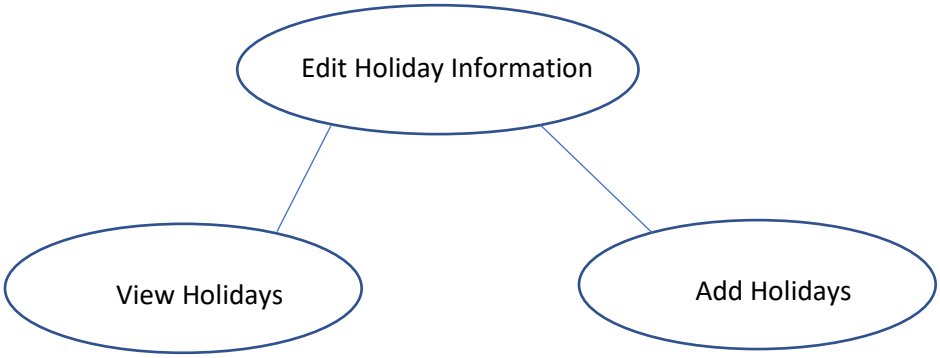
- Show "***Display Main Menu***", "***Edit Holiday Information***", ***"Edit City Population***", ***"View Products by Category***", ***"Analyze revenue for Couches and Sofas***", ***"View Store Revenue by State***", ***"Analyze Sales for Outdoor Furniture on Groundhog Day***", ***"View States with highest volume sold by category***", ***"Calculate revenue by population categories***", ***"Analyze Sales by volume for each Childcare category***", ***"Analyze restaurant impact on sales by categories***", and "***Analyze sales during Advertising Campaigns***" tabs.

- Upon:
  - Click ***Display Main Menu*** button: Jump to the **Main Menu** task.
  - Click ***Edit Holiday Information*** button: Jump to the **Holiday Information Form** task.
  - Click ***Edit City Population*** button: Jump to the **City Population Form** task.
  - Click ***View Products by Category*** button: Jump to the **Report 1** task.
  - Click ***Analyze Revenue for Couches and Sofas*** button: Jump to the **Report 2** task.
  - Click ***View Store Revenue by Year by State*** button: Jump to the **Report 3** task.
  - Click ***Analyze Sales for Outdoor Furniture on Groundhog Day*** button: Jump to the **Report 4** task.
  - Click ***View States with Highest Volume Sold by Category*** button: Jump to the **Report 5** task.
  - Click ***Calculate Revenue by Population Categories*** button: Jump to the **Report 6** task.
  - Click ***Analyze Sales by Volume for each Childcare Category*** button: Jump to the **Report 7** task.
  - Click ***Analyze Restaurant Impact on Sales by Categories*** button: Jump to the **Report 8** task.
  - Click ***Analyze Sales during Advertising Campaigns*** button: Jump to the **Report 9** task.

- Display statistics:
  - the count of stores,
  - count of stores that offer food (have a restaurant, a snack bar, or both),
  - count of stores offering childcare,
  - count of products,
  - count of distinct advertising campaigns.

Holiday Information Form

Task Decomp



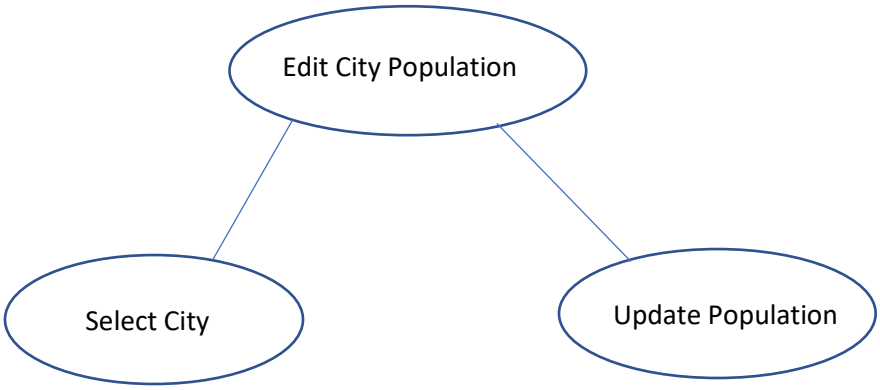
Lock Types: Lookup holiday.holidayName (read-only upon display). Write holiday.holidayName (write upon holiday add).
Number of Locks: 2
Enabling Conditions: Trigger by click from **Main Menu**.
Frequency: Different frequencies
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is needed to separate subtasks for view and add.

Abstract Code

- User clicked on ***Edit Holiday Information*** button from **Main Menu**:
- Upon:
    - Click ***Display Holiday*** button: Query database for current holidays (date and Name) and display them.
    - User will fill in DATE and HOLIDAY NAME input fields.
    - Click ***Add Holiday button***:
        - If Date is valid and does not already exist: Insert new Holiday instance with those values.
        - Otherwise, Display proper warning message.
    - Click ***Main Menu*** button: Jump to the **Main Menu** task.

City Population Form

Task Decomp



Lock Types: Lookup city.population (read-only upon display). Write city.population (write upon population updated).
Number of Locks: 2.
Enabling Conditions: Trigger by click from **Main Menu**.
Frequency: Different frequencies
Consistency (ACID): not critical, order is not critical.
Subtasks: Mother Task is needed to separate subtasks for reads and writes.

Abstract Code

- User clicked on ***Edit City Population*** button from **Main Menu**:
- Upon:
  - Click ***Select City*** button: User select the city they want to update from a drop list; Query database for the City (Name and population) and display them.
  - User will fill in population input fields.
  - Click ***Update Population*** button:
    - If Population is valid: Insert new population instance with those values.
    - Otherwise, Display proper warning message.
  - Click ***Main Menu*** button: Jump to the **Main Menu** task.

Category Report

Task Decomp

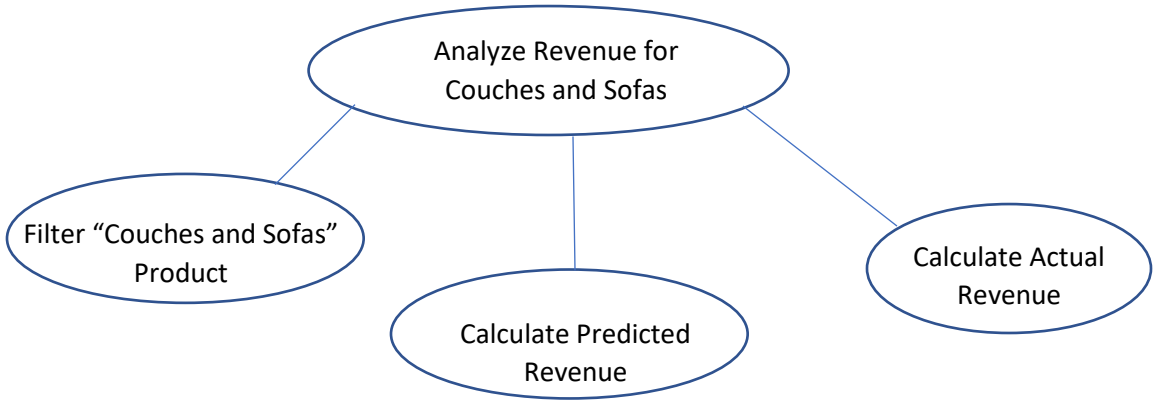
Lock Types: 1 read-only lock of price information from product, 1 read-only lock of the belongTo relationship.
Number of Locks: 2
Enabling Conditions: Trigger by click of ***Category Report*** from **Main Menu**.
Frequency: Low.
Consistency (ACID): This task is read-only, so order is not critical.
Subtasks: No Mother Task is needed.

Abstract code:

- User clicked on the ***Category Report*** button from the **Main Menu**.
- Display **Category Report** screen.
    - Query belongTo relationship to get product category information.
    - Join with Product entity for retail price information.
    - Group by Category name
        - Select total number of Products in category
        - Select minimum retail price in category, 0 if no products
        - Select average retail price in category, 0 if no products
        - Select maximum retail price in category, 0 if no products
    - Display all information sorted by category name ascending.
- User can click on the ***Main Menu*** button to return to the **Main Menu**.

Actual versus Predicted Revenue for Couches and Sofas

Task Decomp



Lock Types: 1 read-only lock of the product entity, 1 read-only lock of the belongTo relationship, 1 read-only lock of the sold relationship, 1 read-only lock of the hasDiscount relationship.
Number of Locks: 4. Several different schema constructs are needed.
Enabling Conditions: Trigger by click of ***Actual versus Predicted Revenue for Couches and Sofas*** from **Main Menu**.
Frequency: Different frequencies
Consistency (ACID): not critical, order is not critical.
Subtasks: Mother Task is needed.

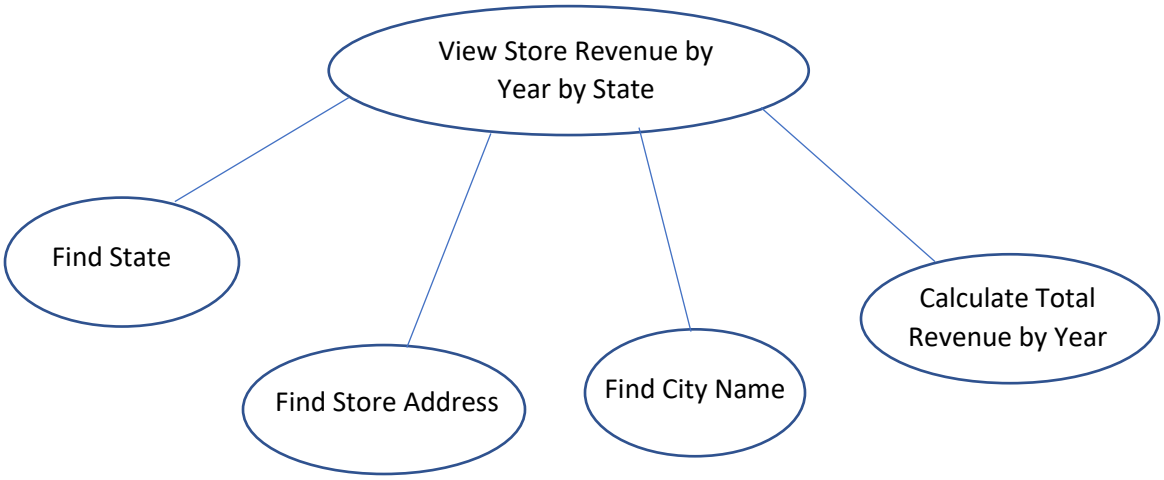
Abstract code:

- User clicked on the ***Actual versus Predicted Revenue for Couches and Sofas*** button from the **Main Menu**.
- Display **Actual versus Predicted Revenue for Couches and Sofas** screen.
    - Query belongTo relationship.
        - Filter for Category name "Couches and Sofas".
        - Join with Product entity
            - Select product.PID, product.name and product.retailPrice.
        - Join with sold relationship for sales information.
            - Group by Product
            - Select total number units sold
            - Calculate predicted revenue from 0.75 units sold * retail price
        - Join with hasDiscount relationship for discount information.
            - Group by sold relationship
            - Select total number units sold at retail and discount price by comparing sale date and discount date
            - Calculate actual revenue from units sold at retail and discount
    - Calculate predicted and actual revenue difference

    - Display predicted and actual revenue differences greater than $5,000 sorted in descending order
- User can click on the ***Main Menu*** button to return to the **Main Menu**.

Store Revenue by Year by State

Task Decomp



Lock Types: Lookup store.storeID, store.streetAddress, city.state, city.name, date.year, sold.quantity, discount.discountPrice, product.retailPrice from store, city, date, sold, discount, and product, all are Read-only.
Number of Locks: 6. Several different schema constructs are needed
Enabling Conditions: Trigger by clicking on ***Store Revenue by Year by State*** button from Main Menu.
Frequency: Different frequencies. Three "find" subtasks have same frequency.
Consistency (ACID): not critical, order is not critical.
Subtasks: Mother Task is needed.

Abstract Code

- User clicked on ***Store Revenue by Year by State*** button from **Main Menu**:
- Run the **Store Revenue by Year by State** task.
- User select $state from a drop-down box.
- For each state, find all available stores from sold by sold.storeID=store.storeID & store.cityID=city.cityID & city.state = $state.
- For each store, do:
  - Find $address from store.streetAddress;
  - Find $city where the store locates by city.name when store.cityID = city.cityID;
  - Find all available years from date.year when store.storeID=sold.storeID & sold.dateID=date.dateID.
    - For each $year, calculate the total revenue of the store ($totalRevenue_per_store) by summing the total revenue of each product ($totalRevenue_per_product), the revenue of product is calculated as follows
      If the product is on sale:
      $totalRevenue_per_product = sold.quantity * discount.discountPrice

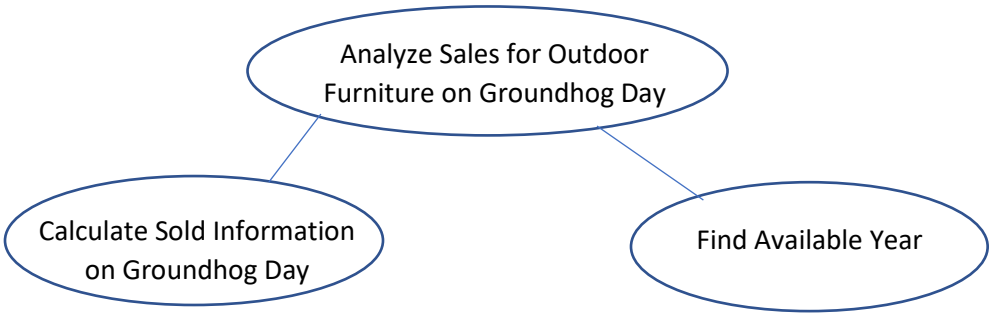
Else:

$totalRevenue_per_product = sold.quantity * product.retailPrice

- Display $storeID, $address, $city, $year, and $totalRevenue_per_store, first arranged by year in ascending order, and then by revenue in descending order.
- click ***Main Menu*** button, return to **Main Menu**.

Outdoor Furniture on Groundhog Day

Task Decomp



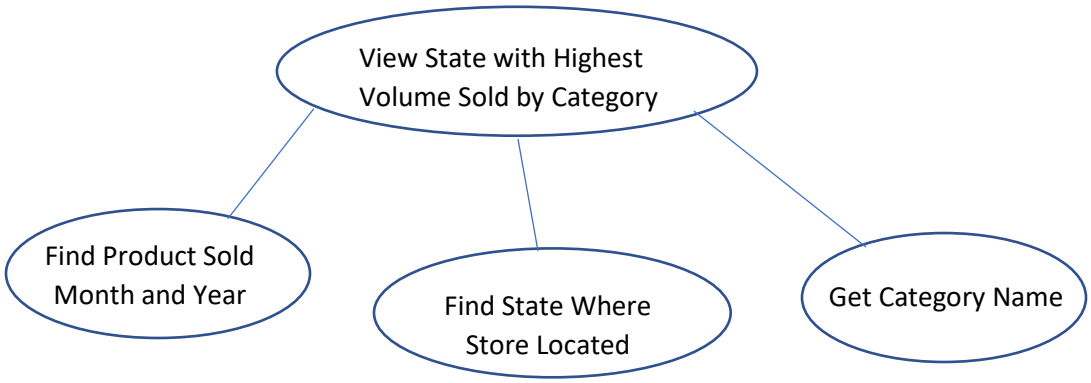
Lock Types: Lookup date.year, date.month, date.day, sold.quantity, product.PID, belongTo.categoryName from date, sold, product, and belongTo, all are Read-only.
Number of Locks: 4. Several different schema constructs are needed
Enabling Conditions: Trigger by clicking on ***Outdoor Furniture on Groundhog*** Day button from Main Menu.
Frequency: Different frequencies.
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is needed.

Abstract Code

- User clicked on ***Outdoor Furniture on Groundhog Day*** button from **Main Menu**:
- Run the **Outdoor Furniture on Groundhog Day** task.
- Find all available years from date.year when sold.dateID=date.dateID, arranging years in ascending order.
- For each $year, do:
  - Calculating the total number of items sold ($total_quantity_per_year) in the outdoor furniture category by summing sold.quantity when sold.PID=product.PID & product.PID = belongTo.PID & belongTo.categoryName = “Outdoor furniture”;
  - Calculating the average number of units sold per day ($avg_quantity_per_day) by $total_quantity_per_year/365;
  - Calculating the total number of units sold on Groundhog Day ($total_quantity_on_GroundhogDay) by summing sold.quantity when sold.PID = product.PID & product.PID = belongTo.PID & belongTo.categoryName = “Outdoor furniture” & sold.dateID = date.dateID & date.month = 2 & date.day = 2;
  - Setting $significance = “TRUE” if $total_quantity_on_GroundhogDay/$avg_quantity_per_day>=2.
- Display $year, $total_quantity_per_year, $avg_quantity_per_day, $total_quantity_on_GroundhogDay, and $significance, arranging by year in ascending order.
- click ***Main Menu*** button, return to **Main Menu**.
  User selects next action from choices in **Main Menu**.

View State with Highest Volume Sold by Category

Task Decomp



Lock Types: read-only sold.dateID, date.dateID, date.year, date.month, store.storeID, city.state, product.PID, category.categoryName from city, date, store, product, category.
Number of Locks: 5. Several different schema constructs are needed
Enabling Conditions: Trigger by click ***View States with highest volume sold by Category*** from **Main Menu**.
Frequency: Low. Same frequency for all Subtasks
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is needed.

Abstract Code

- User clicked on ***View States with highest volume sold by Category*** link from **Main Menu**
- Display **State with highest volume sold by Category** screen
- Join query the database for years and months represented in Sold relationship use this to populate transaction month and year drop down (sold.dateID = date.dateID and date.month = $month and date.year = $year). Sort month and year ascending
- Users choose month and year from available Date in the dropdown list.
- **Upon:**
    - ***View Report*** button click:
        - Clear previous display
        - Read the user's selected month/year from the dropdown list in the **State with highest volume sold by Category** screen
        - Join query the database to find city.states in relationship hasStore with the sold instances' stores (sold.store.storeID) and the total number of unit sold (SUM(sold.quantity)). All of the products sold will be checked if it was sold during the selected month by using sold.product.PID in relationship belongTo with category.Name based on sold.date.month and

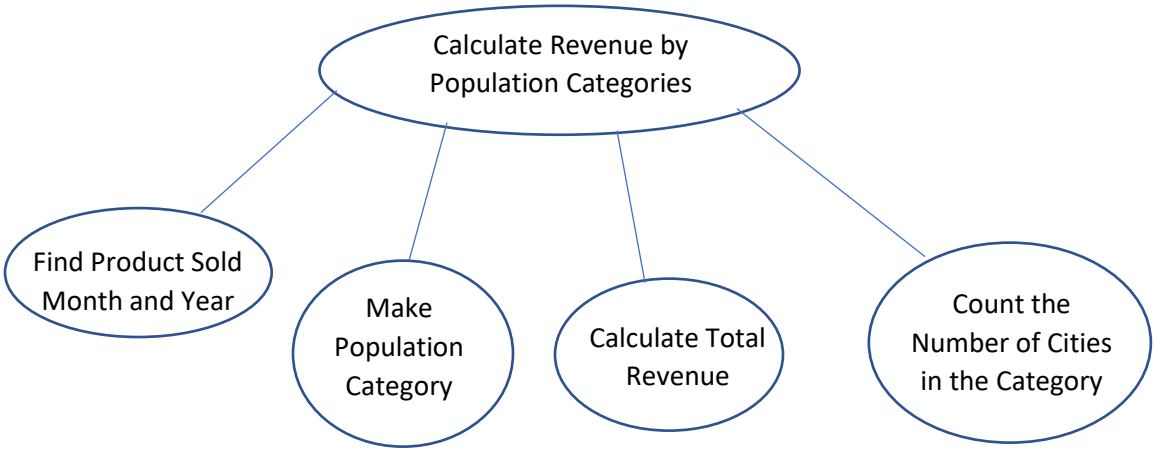
sold.date.year. All will be group by state and ordered by category.Name ascending order.

- For each category.name, the report display the category.name followed by city.state and the highest number of units sold for that category in a new line.
    - If there are no unites sold of that category during that month/year, display only the category.name and “No units sold”

- User click ***Main Menu*** button: return to **Main Menu**.

Calculate Revenue by Population Categories

Task Decomp



Lock Types: look up city.cityID, city.Population, date.dateID, product.PID, product.retailPrice, sold.quantity from city, date, product, sold, all are read-only
Number of Locks: 4. Several different schema constructs are needed
Enabling Conditions: Trigger by click ***Calculate revenue by population categories*** from **Main Menu**.
Frequency: Low.
Consistency (ACID): Not critical. Population change tracking is not required. The task picks up the current population of each city. Lookup must be done first followed by any number of lookups.
Subtasks: Mother Task is not required

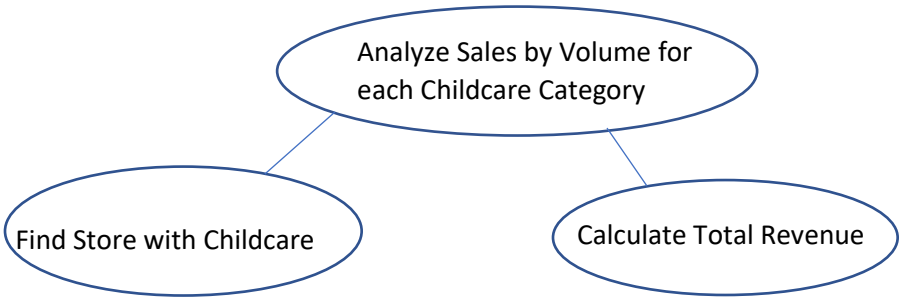
Abstract Code

- User clicks on ***Calculate revenue by population categories*** from **Main Menu**
- Joint query the database for years represented in Sold relationship use this to populate transaction year (sold.dateID = date.dateID and date.year = $year). Allow user to input which $year need to be report. Ordered by years ascending order.
- Calculate the $TotalAnnualRevenue for each of four population categories ("Small" < 3,700,000; "Medium" >= 3,700,000 and < 6,700,000; "Large" >= 6,700,000 and < 9,000,000; and "Extra Large" >= 9,000,000) and for all sold instances each year (based on date.year). Ordered by population categories ascending order.
    - To calculate total sales for a store (store.storeID) within a year, look for each sold instance for that store and find the related product.PID and date.year
        - Exclude the sale if date.year is not within the given $year
        - If the product.PID has relationship hasDiscount:
            - Define $TOTALSALE = hasDiscount.discountPrice * sold.quantity
        - Otherwise:
            - Define $TOTALSALE = product.retailPrice * sold.quantity

        - Check if the store.storeID has relationship hasStore within city whose city.population is within the categories.
            - If so, count the sale for that year and population category.
            - If not, exclude the sale for that year and population category.
    - Calculate total annual avenue of each store of the population category for each year = sum of all $TOTALSALE for each store during that year
    - Calculate the number of cities within the category by counting city.Name where store hasStore in city that have sold instances on the date within date.year
    - Sort the distinct years represented in any of the sold instances in ascending order. Display the row headings of the grid as "Small", "Medium", "Large", and "Extra Large". Display the respective calculated $TotalAnnualRevenue in each cell.
- Upon:
    - User click ***Main Menu*** button click to display **Main Menu**.

Childcare Sales Volume

Task Decomp



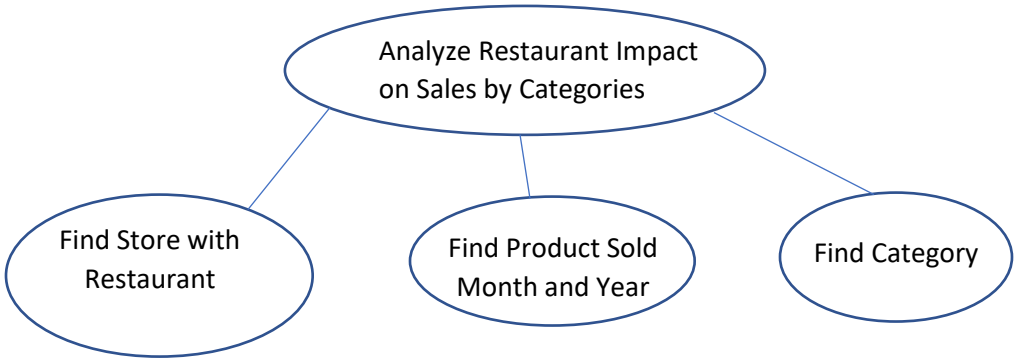
Lock Types: Lookup store.storeID, store.childcareLimit, date.year, sold.quantity, discount.discountPrice, product.retailPrice from store, date, sold, discount, and product, all are Read-only.
Number of Locks: 5. Several different schema constructs are needed
Enabling Conditions: Trigger by ***clicking on View Analyze Sales by volume for each Childcare category*** from Main Menu.
Frequency: Different frequency.
Consistency (ACID): Must find store with childcare first.
Subtasks: Mother Task is needed.

Abstract Code

- User clicked on ***View Analyze Sales by volume for each Childcare category*** button from **Main Menu**:
- Run the ***Analyze Sales by volume for each Childcare category*** task.
- User select $childcareLimit from a drop-down box.
- For each childcare limit, find all available stores from sold by sold.storeID=store.storeID & store.childcareLimit = $ childcareLimit.
- For each store, do:
  - Find the last available years from date.year when store.storeID=sold.storeID & sold.dateID=date.dateID.
    - For each $month, calculate the total revenue of the store ($totalRevenue_per_store) by summing the total revenue of each product ($totalRevenue_per_product), the revenue of product is calculated as follows
      If the product is on sale:
      $totalRevenue_per_product = sold.quantity * discount.discountPrice
      Else:
      $totalRevenue_per_product = sold.quantity * product.retailPrice
- Display $storeID, $ childcareLimit, $month, $year, and $totalRevenue_per_store, first arranged by month in ascending order, and then by childcareLimit in descending order.
- click ***Main Menu*** button, return to **Main Menu**.

Restaurant Impact on Category Sales

Task Decomp



Lock Types: Look up sold.dateID, date.dateID, date.year, date.month, Store.storeID, City.State, product.PID, category.categoryName from store, date, sold, discount, and product, all are Read-only.
Number of Locks: 5. Several different schema constructs are needed
Enabling Conditions: Trigger by clicking on View Analyze Restaurant Impact on Sales by Categories from Main Menu.
Frequency: Low. Same frequency for all subtasks
Consistency (ACID): Must find store with restaurant first.
Subtasks: Mother Task is needed.

Abstract Code

- User clicked on ***View Analyze Restaurant Impact on Sales by Categories*** button from **Main Menu**:
- Run the ***Analyze Restaurant Impact on Sales by Categories*** task.
- Join query the database for years and months represented in Sold relationship use this to populate transaction month and year drop down (sold.dateID = date.dateID and date.month = $month and date.year = $year). Sort month and year ascending
- Users choose month and year from available Date in the dropdown list.
- **Upon:**
    - ***View Report*** button click:
        - Clear previous written lines
        - Read the user's selected month/year from the dropdown list in the ***Restaurant Impact on Sales by Categories Screen***.
        - Query database for a sorted (ascending) list of all category.categoryName
            - For each category.categoryName (in sorted order):
                - Query the database to find store.hasRestaurant the sold instances' stores (sold.store.storeID) and the total number of unit sold (SUM(sold.quantity)). All of the products sold

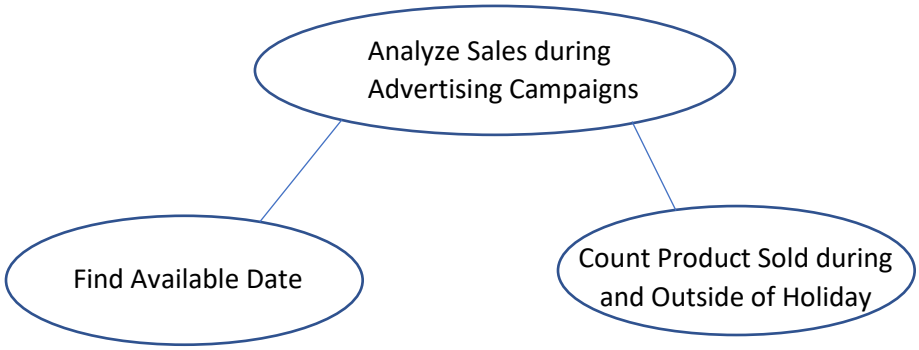
will be checked if it was sold during the selected month by using sold.product.PID in relationship belongTo with category.Name based on sold.date.month and sold.date.year. All will be group by store type (store.hasRestaurant) and ordered by category.Name ascending order.

- For each category.categoryName, the report display the category.categoryName followed by hasRestaurant and the number of units sold for that category in a new column.
  - If there are no unites sold of that category during that month/year, display only the category.categoryName followed by hasRestaurant and “No units sold”

- User clicks ***Main Menu*** button click run Display Main Menu task.

Advertising Campaign Analysis

Task Decomp



Lock Types: Look up sold.dateID, date.dateID, date.year, date.month, Store.storeID, product.PID, product.name from store, date, sold, discount, and product, all are Read-only.
Number of Locks: 5. Several different schema constructs are needed
Enabling Conditions: Trigger by clicking on ***View Analyze Sales during Advertising Campaigns*** from **Main Menu**.
Frequency: Different frequency. Count product sold has higher frequency
Consistency (ACID): Not critical, order is not critical.
Subtasks: Mother Task is needed.

Abstract Code

- User clicked on ***View Analyze Sales during Advertising Campaigns*** button from **Main Menu**:
- Run the ***Analyze Sales during Advertising Campaigns*** task.
- Join query the database for years and months represented in Sold relationship use this to populate transaction month and year drop down (sold.dateID = date.dateID and date.month = $month and date.year = $year). Sort month and year ascending
- Users choose month and year from available Date in the dropdown list.
- **Upon:**
    - ***View Report*** button click:
        - Clear previous written lines
        - Read the user's selected month/year from the dropdown list in the ***Analyze Sales during Advertising Campaigns Screen.***
        - Query database for a sorted (ascending) list of all product.PID
            - For each product.PID (in ascending order):
                - Query the database to find the product.name the sold instances' stores (sold.store.storeID) and the total number of unit sold (SUM(sold.quantity)). All of the products sold will be checked if it was sold during the selected month by using sold.product.PID based on sold.date.month and

sold.date.year that has relationship hasAdvertisingCampaign. All will be group by date and ordered by ascending order.

- For each product.ID, the report displays the product.ID followed by product.name and the number of units sold during date that has relationship hasAdvertisingCampaign in a new column called $ SOLDDURINGCAMPAIGN. The number of units sold during date that does not has relationship hasAdvertisingCampaign are shown in a new column called $SOLDOUTSIDECAMPAIGN.
  - Define $SOLDDURINGCAMPAIGN = SUM(sold.quantity.date from hasAdvertisingCampaigns relationship)
  - Define $SOLDOUTSIDECAMPAIGN = SUM(sold.quantity.date that does not have hasAdvertisingCampaigns relationship)
- If there are no unites sold of that category during that month/year, display only the product.ID followed by product.name and “No units sold”
- Calculate the difference between Sold During Campaign and Sold Outside Campaign for each product.ID. Display the respective calculated $Difference in each cell.

- User click ***Main Menu*** button to display **Main Menu** task.