//Report 1 (Don’t include this title in the final pdf report)

View Product by Category Report

Task Decomp



**Lock Types**: 2 Read-only 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.

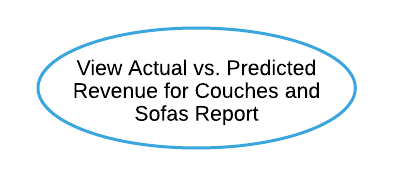
Abstract Code

* 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.

//Report 2 (Don’t include this title in the final pdf report)

View Actual vs. Predicted Revenue for Couches and Sofas Report

Task Decomp



**Lock Types**: 5 Read-only 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).
  + Get PID, Product\_Name, and Retail\_Price data for each product of Couches and Sofas category (from the PRODUCT table).
  + 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.
  + Find actual revenue in one day 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 and sort revenue difference in descending order.
* When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

//Report 4 (Don’t include this title in the final pdf report)

View Groundhog Day Outdoor Furniture Report

Task Decomp



**Lock Types**: 3 Read-only 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.

//Report 8 (Don’t include this title in the final pdf report)

View Restaurant Impact on Category Sales Report

Task Decomp



**Lock Types**: 4 Read-only 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 assigned (from CATEGORY and PRODUCT tables).
  + For each category:
    - Get Store\_Number data for all stores that sell products of that category (from STORE and PRODUCT tables).
    - If store Has\_Restaurant (from the STORE table) is TRUE:
      * Display store type as “Restaurant”.
      * Find total quantity sold by aggregating all Quantity (from the SALE table) for all stores with a restaurant
    - Else:
      * Display the store type as “Non-Restaurant”.
      * Find total quantity sold by aggregating all Quantity (from the SALE table) for all stores without a restaurant
* Group by Category\_Name ascending and with “Non-Restaurant” store data listed first.
* When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

//Report 9 (Don’t include this title in the final pdf report)

View Advertising Campaign Analysis Report

Task Decomp



**Lock Types**: 5 Read-only 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 number of products 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 number of products 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 that 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.