

S&E Technology Superstore Data Warehouse Phase I

Project Report

Team024

[jchen857 jhu369 twu76 yshan34]



Table of Contents

Section 1 — Data Type.....	3
Section 2 — Business Logic Constraints.....	4
Section 3 — Task Decomposition with Abstract Code	5
Main Menu	5
Edit Holiday	7
Edit Manager	8
Add Manager.....	9
Remove Manager	10
Assign Manager.....	11
Unassign Manager.....	12
Edit Population.....	13
View Manufacturer’s Product Report	14
View Category Report.....	15
View Actual versus Predicted Revenue for GPS units Report.....	16
View Store Revenue by Year by State	18
View Air Conditioners on Groudhog Day	19
View State with Highest Volume for each Category	20
View Revenue by Population.....	21

Section 1 — Data Type

store

Attribute	Data Type	Nullable
store number	CHAR(10)	Not Null
phone number	String	Null
street address	VARCHAR(100)	Not Null

city

attribute	Data Type	Nullable
city	VARCHAR(50)	Not Null
state	VARCHAR(20)	Not Null
population	Integer	Not Null

manager

attribute	Data Type	Nullable
email	VARCHAR(50)	Not Null
name	VARCHAR(50)	Not Null

product

attribute	Data Type	Nullable
PID	CHAR(20)	Not Null
name	VARCHAR(50)	Not Null
retail price	Decimal(10,2)	Not Null
max discount(%)	Decimal(2,0)	Null

manufacturer

Attribute	Data Type	Nullable
name	VARCHAR(50)	Not Null

category

Attribute	Data Type	Nullable
name	VARCHAR(50)	Not Null

date

Attribute	Data Type	Nullable
month	Decimal(2,0)	Not Null
day	Decimal(2,0)	Not Null
year	Decimal(4,0)	Not Null
holiday name	VARCHAR(100)	Null

sales record

Attribute	Data Type	Nullable
quantity	Integer	Not Null

onSale

Attribute	Data Type	Nullable
sale price	Decimal(10,2)	Not Null

Section 2 — Business Logic Constraints

Entities:

manager

- manager will become inactive if they are not be assigned to any store or if they are not an S&E employee any more.
- The data of manager cannot be deleted unless they have been unassigned from all of their stores.

Attributes:

email - should be an email format

store

Attributes:

phone number - the number format should be country code + phone number

city

Attributes:

state - should be selected from a state list

population - cannot be negative

date

Attributes:

month - select from 1-12

Project Phase 1 | CS 6400 – Spring 2019 | Team 024

day - Month 1, 3, 5, 7, 8, 10, 12 shall have 31 days, Month 4, 6, 9, 11 shall have 30 days, Month 2 shall have 29 days when in leap years otherwise 28 days.

holiday name - If a date is a holiday, there will be a holiday name associated with it. If a date is not a holiday, holiday name will be null.

product

Attributes:

max discount - a product cannot be placed on sale if maximum discount is set 0%.

retail price - retail price will be used as price in a specific day if that product is not set on Sale in that day, otherwise, sale price will be used.

Relations:

sales records

Attributes:

Quantity - cannot be negative

onSale

- All stores sell the same product at the same price If a product is on sale.

Attributes:

sale price

- sale price of a product cannot be higher than regular retail price.
- sale price of a product cannot be smaller than the price with maximum discount applied. (Only applicable, when the manufacture has specified maximum discount.)
- sale price of a product can be different only when in different days. (Different sale prices of a product are not allowed if in the same day.)
- sale price of a product cannot be less than 10% of regular retail price if manufacture does specify maximum discount. (No product can be discounted more than 90% of regular retail price if no maximum discount is specified by the manufacture.)

Section 3 — Task Decomposition with Abstract Code

Main Menu



Main Menu

Task Decomposition

Lock Type: Read-only on [Store](#), [Manager](#), [Product](#), and [Manufacturer](#) table

Number of Locks: Multiple

Table of Contents

Revised 02/10/2019

Project Phase 1 | CS 6400 – Spring 2019 | Team 024

Enabling Conditions: None

Frequency: User login and menu option have the same frequency

Consistency (ACID): Trigger by successful login which is taken care by security team.

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

Abstract Code

- Find the **store** using **store** [store number]; Display the count of store (count(**store**[store number])).
- Find the **manufacturer** using **manufacturer** [name]; Display the count of manufacturer (count (**manufacturer** [name])).
- Find the **product** using **product** [PID]; Display the count of products (count (**product** [PID])).
- Find the **manager** using **manager**[email]; Display the count of manager (count (**manager** [email])).
- Show “*Edit Holiday*”, “*Edit Manager*”, “*Edit Population*”, “*View Manufacturer’s Product Report*”, “*View Category Report*”, “*View Actual versus Predicted Revenue for GPS units Report*”, “*View Store Revenue by Year by State Report*”, “*View Air Conditioners on Groundhog Day Report*”, “*View State with Highest Volume for each Category Report*”, “*View Revenue by Population Report*”, and “*Log out*” tabs.
- Upon:
 - Click *Edit Holiday* button – Jump to the *Edit Holiday* task.
 - Click *Edit Manager* button – Jump to the *Edit Manager* task.
 - Click *Edit Population* button – Jump to the *Edit Population* task.
 - Click *View Manufacturer’s Product Report* button – Jump to the *View Manufacturer’s Product Report* task.
 - Click *View Category Report* button – Jump to the *View Category Report* task.
 - Click *View Actual versus Predicted Revenue for GPS units Report* button – Jump to the *View Actual versus Predicted Revenue for GPS units Report* task.
 - Click *View Store Revenue by Year by State Report* button – Jump to the *View Store Revenue by Year by State Report* task.
 - Click *View Air Conditioners on Groundhog Day Report* button – Jump to the *View Air Conditioners on Groundhog Day Report* task.
 - Click *View State with Highest Volume for each Category Report* button – Jump to the *View State with Highest Volume for each Category Report* task.
 - Click *View Revenue by Population Report* button – Jump to the *View Revenue by Population Report* task.
 - Click *View Revenue by Population Report* button – Jump to the *View Revenue by Population Report* task.
 - Click *Log out* button – Go back to **Login** form.

Edit Holiday



Edit Holiday

Task Decomposition

Lock Type: Read and write on [Date](#) table

Number of Locks: Single

Enabling Conditions: user successfully login and clicked *Edit Holiday*

Frequency: Low. Depends on the business requirement.

Consistency (ACID): is not critical, even if a report is looking for the holiday while date is being edited

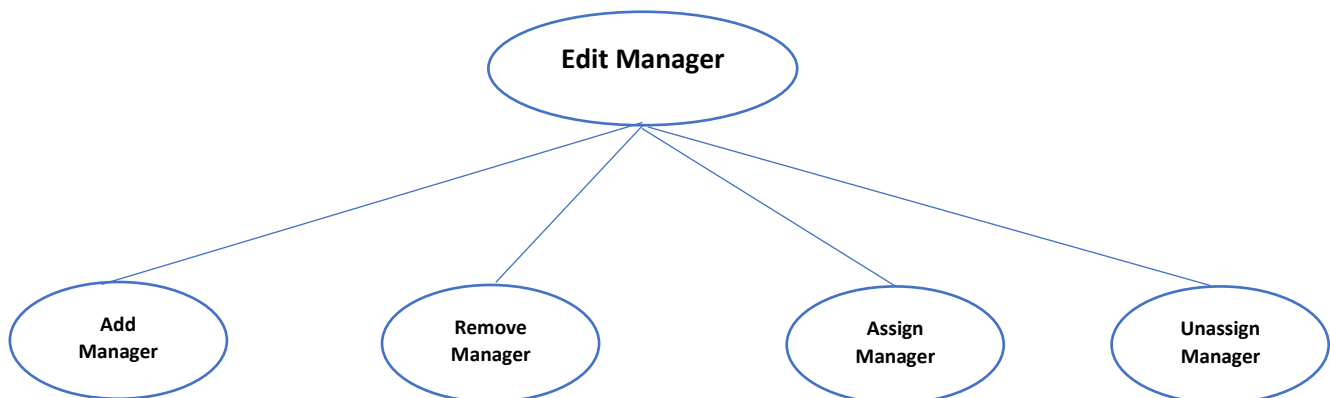
Subtasks: Mother task is not needed. No decomposition needed

Abstract Code

- User clicked *Edit Holiday* button from Main Menu.
- Go to the default page for Edit Holiday
 - There is a *View* button for user to see the date holiday details, an *Add* button for user to add holidays, and an *Update* button for user to update holidays.
- When *View* button is pushed, pop up a window with a drop-down menu for month filed, a drop-down menu for day field and a holiday name field. The month field and day filed are set to be “01” by default. The holiday name filed is empty. There are two buttons, *Search* and *Cancel* button.
 - When *Search* button is pushed, display [date](#) [date] and [date](#) [holiday name] where [date](#) [date] in date or date range is selected by user.
 - Perform the query: select date and holiday name from [Date](#) table where holiday name is not null and not blank.
 - Display the date and holiday names for the results of the above query.
 - If user push *Cancel* button, return to the default page for Edit Holiday.
- When *Add* button is pushed, pop up a window with a drop-down menu for month filed, a drop-down menu for day field and a holiday name field. The month field and day filed are set to be “01” by default. The holiday name filed is empty.
- User can fill-in the information for the additional holiday, ‘holiday name’. There are two buttons, *Save* and *Cancel* button.
 - If user pushes *Save* button:
 - If user enter empty holiday name:
 - Stay at Edit Holiday page with error message, ‘holiday name cannot be empty’.

- Else:
 - If the holiday name already exists for this date (`date[date] == 'date'` and `date[holiday name] == 'holiday name'`):
 - Pop a window saying, “This holiday is already in the database” and go back to the default page.
 - If the holiday name doesn't exist for this date (`date[date] == 'date'` and `date[holiday name] <> 'holiday name'`):
 - Set the holiday name for this date to be the concatenate of the original holiday name and the new holiday name.
- After inserting, return to the default page for **Edit Holiday**. Now the Date table has been updated and the list of holidays would not have the original holiday but the user-entered holiday.
 - If user push ***Cancel*** button, return to the default page for **Edit Holiday**
- When ***Update*** button is pushed, pop up a window with a drop-down menu for month filed, a drop-down menu for day field and a holiday name field. The month field and day filed are set to be “01” by default. The holiday name filed is empty. User can fill-in the information for the additional holiday, ‘holiday name’. There is a ***Save*** button and a ***Cancel*** button too.
 - If user pushes ***Save*** button, set the holiday name for this date to be the user-entered holiday name (`date[holiday name] = 'holiday name'`).
 - If user-entered holiday name is blank, it means holiday name is removed.
 - After updating, return to the default page for **Edit Holiday**.
 - If user push ***Cancel*** button, return to the default page for **Edit Holiday**
- When ***Main Menu*** button is pushed:
 - Go to the **Main Menu** form.
- When ***Log out*** button is pushed:
 - Go to the **Login** form.

Edit Manager



Task Decomposition

Lock Type: Read and write on [Manger](#) table

Number of Locks: Single

Enabling Conditions: user successfully login and clicked *Edit Manager*

Frequency: Low. Depends on the business requirement.

Consistency (ACID): is not critical, even if a report is looking for the manager information while manager is being edited

Subtasks: can be decomposed to **Assign/Unassign Manager** tasks and **Add/Remove Manager** task. Must be done separately.

Abstract Code

- User clicked on *Edit Manager* button from Main Menu.
- When *View* button is pushed:
 - User enters any of the following fields: *manager name* ('manager name'), *manager email* ('email address'), and *store* ('store number').
 - Find [managers](#) that match the user-entered information using [manager](#) [manager name], [manager](#) [email address], and [store](#)[store number]. Ignore blank fields.
 - Display manager[name], and manager [email address] in [manager](#) table and relevant store[store number] in [store](#) table from [manager](#) – [store](#) relationship.
- When *Add Manager* button is pushed, jump to **Add Manager** Task.
- When *Remove Manager* button is pushed, jump to **Remove Manager** Task.
- When *Assign Manager* button is pushed, jump to **Assign Manager** Task.
- When *Unassign Manager* button is pushed, jump to **Unassign Manager** Task.
- When *Main Menu* button is pushed:
 - Go to the Main Menu form.
- When *Log out* button is pushed:
 - Go to the Login form.

Add Manager



Add Manager

Task Decomposition

Lock Type: Read and write on [Manger](#) table

Number of Locks: Single

Project Phase 1 | CS 6400 – Spring 2019 | Team 024

Enabling Conditions: user successfully login and clicked *Add Manager* in Edit Manager form

Frequency: Low. Depends on the business requirement.

Consistency (ACID): is not critical, even if a report is looking for the manager information while manager is being edited

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

Abstract Code

- User clicked on *Add Manager* button from Edit Manager.
- User enters *manager name* ('manager name'), and *manager email* ('email address') input fields.
- When *Save* button is pushed:
 - If 'manager name' is blank:
 - Stay on Add Manager page with error message, 'Please fill in manager's name'.
 - If 'email address' does not follow the standard format:
 - Stay on Add Manager page with error message, 'Manager email is not in standard format'.
 - If 'email address' already exists:
 - Stay on Add Manager page with error message, 'Manager email already exists'.
 - Else:
 - Add 'manager name', and 'email address' into *manager* table with success message.
- When *Main Menu* button is pushed:
 - Go to the Main Menu form.
- When *Log out* button is pushed:
 - Go to the Login form.

Remove Manager

Task Decomposition

Lock Type: Read and write on *Manager* table

Number of Locks: Single

Enabling Conditions: user successfully login and clicked *Remove Manager* in Edit Manager form

Frequency: Low. Depends on the business requirement.

Table of Contents



Remove Manager

Project Phase 1 | CS 6400 – Spring 2019 | Team 024

Consistency (ACID): is not critical, even if a report is looking for the manager information while manager is being edited

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

Abstract Code

- User clicked on **Remove Manager** button from **Edit Manager**.
- User enters *manager email* ('email address') input field.
- When **Save** button is pushed:
 - If 'manager email' is blank:
 - Stay on **Remove Manager** page with error message, 'Please fill in the manager's email that you would like to remove'.
 - If 'manager email' not in **manager** [manager email]:
 - Stay on **Remove Manager** page with error message, 'Manager is not in system'.
 - If 'manager email' in **manager** [manager email] and 'manager email' in **Manager – Store** relationship, which means this manager is still assigned to store as an **active manager**:
 - Stay on **Remove Manager** page with error message, 'Manager needs to be unassigned first'.
 - Else:
 - Remove **manager** [manager name], and **manager** [email address] based on 'manager email' with success message.
- When **Main Menu** button is pushed:
 - Go to the **Main Menu** form.
- When **Log out** button is pushed:
 - Go to the **Login** form.

Assign Manager

Task Decomposition

Lock Type: Read and write on **Manager** table

Number of Locks: Single

Enabling Conditions: user successfully login and clicked **Assign Manager** in **Edit Manager** form

Frequency: Low. Depends on the business requirement.

Consistency (ACID): is not critical, even if a report is looking for the manager information while manager is being edited

Table of Contents



Revised 02/10/2019

Project Phase 1 | CS 6400 – Spring 2019 | Team 024

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

Abstract Code

- User clicked on *Assign Manager* button from Edit Manager.
- User enters *manager email* ('manager email') and *store* ('store number') input fields.
- When *Save* button is pushed:
 - If 'manager email' not in *manager* [manager email] or 'manager email' is blank:
 - Stay on Assign Manager page with error message, 'manager cannot be found'.
 - If 'store number' not in *store* [store number] or 'store number' is blank:
 - Stay on Assign Manager page with error message, 'store cannot be found'.
 - If a record can be found in *Manage- Store* relationship with the manager email and store number:
 - Stay on Assign Manager page with success message, 'Manger is already assigned to a store'.
 - Else:
 - Add the record of 'manager email' and 'store number' in *Manage - Store* relationship with success message, 'Manger is already assigned to store'.
 - This manager will be marked as *active manager*.
- When *Main Menu* button is pushed:
 - Go to the Main Menu form.
- When *Log out* button is pushed:
 - Go to the Login form.

Unassign Manager

Task Decomposition



Lock Type: Read and write on *Manager* table

Number of Locks: Single

Enabling Conditions: user successfully login and clicked *Unassign Manager* in Edit Manager form

Frequency: Low. Depends on the business requirement.

Consistency (ACID): is not critical, even if a report is looking for the manager information while manager is being edited

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

Table of Contents

Revised 02/10/2019

Abstract Code

- User clicked on *Unassign Manager* button from **Edit Manager**.
- User enters *manager* ('manager email') and *store* ('store number') input fields.
- When *Save* button is pushed:
 - If 'manager email' is blank:
 - Stay on **Edit Manager** page with error message, 'Please fill in manager's email'.
 - If 'store#' is blank:
 - Stay on **Edit Manager** page with error message, 'Store cannot be found'.
 - If no record can be found in *Manage - Store* relationship with the manager email and specific store number:
 - Stay on **Edit Manager** page with error message, 'The manager is not assigned to the store.'
 - Else:
 - Remove the records of the manager email and store# from the *Manage – Store* relationship with success message 'The has been successfully unassigned from the store'.
 - If this manager does not have any relation with store, it will be marked as *inactive manager*.
- When *Main Menu* button is pushed:
 - Go to the **Main Menu** form.
- When *Log out* button is pushed:
 - Go to the **Login** form.

Edit Population



Edit Population

Task Decomposition

Lock Type: Read and write on *City* table

Number of Locks: Single

Enabling Conditions: user successfully login and clicked *Edit Population*

Frequency: Low. Depends on the business requirement.

Consistency (ACID): is not critical, even if a report is looking for the city while population is being edited

Subtasks: Mother task is not needed. No decomposition needed

Abstract Code

Table of Contents

Revised 02/10/2019

Project Phase 1 | CS 6400 – Spring 2019 | Team 024

- User clicked on **Edit Population** button from Main Menu.
- When **View** button is pushed:
 - User enters any of the following fields: *city name* ('city name'), and state name ('state').
 - Find **city** using **city** [city name], and **city**[state]. Ignore blank fields.
 - Display **city** [city name], **city** [state], and **city**[population] in **city** table.
- When **Edit** button is pushed:
 - User enters *city name* ('city name') and *population* ('population') input fields.
 - When **Save** button is pushed:
 - If 'city name' not in **city** [city name]:
 - Stay on Edit Population page with error message, 'city cannot be found'.
 - If 'city name' is blank:
 - Stay on Edit Population page with error message, 'city name cannot be blank'.
 - If 'population' does not follow the standard format or is blank:
 - Stay on Edit Population page with error message, 'population is incorrect'.
 - Else:
 - Lookup **city** using **city** [city name] and update population using **city**[population] with success message.
- When **Main Menu** button is pushed:
 - Go to the Main Menu form.
- When **Log out** button is pushed:
 - Go to the Login form.

View Manufacturer's Product Report

Task Decomposition

Lock Type: Read only on **Manufacturer**, **Product**, **Store** tables

Number of Locks: Multiple

Enabling Conditions: user successfully login and clicked **View Manufacturer's Product Report**

Frequency: Low. Depends on the business requirement.

Consistency (ACID): is not critical.

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



Table of Contents

Revised 02/10/2019

Abstract Code

- User clicked on **View Manufacturer's Product Report** button from **Main Menu**.
- Run the **View Manufacturer's Product Report**: query for information about products condition for each manufacturer.
- Find the **manufacturer** using **manufacturer** [name].
- Find the relevant **product** for each **manufacturer** using **product** [PID] and **manufacturer** [name].
- Count the number of products for each manufacturer (count (**product** [PID]))
- Compute the average, minimum and maximum retail price for each manufacturer.
- Display the manufacturer's name, total number of products offered by the manufacturer, average retail price, minimum retail price and maximum retail price for each manufacture; Sort by highest average retail price and only list the top 100 manufacturers. Each manufacturer has a **Detail** button.
- If the user pushes Detail button for a specific manufacturer (target manufacturer), display the manufacturer's name, maximum discount, and all the information mentioned above for this manufacturer. Then perform the following query:
 - Filter the products from the **Product** table by the target manufacturer as target products
 - Find the **category**/categories for the target products. If a product has multiple categories, concatenate them as one string.
 - Display the product ID, name, retail price and **category**/categories for each product with the target manufacturer; Oder by retail price from high to low.
- When **Main Menu** button is pushed:
 - Go to the **Main Menu** form.
- When **Log out** button is pushed:
 - Go to the **Login** form.

View Category Report

Task Decomposition

Lock Type: Read only on **Manufacturer, Product, Store, Category** tables

Number of Locks: Multiple

Enabling Conditions: user successfully login and clicked **View Category Report**

Frequency: Low. Depends on the business requirement.

Consistency (ACID): is not critical.

Subtasks: Mother task is not needed.

Table of Contents



View Category
Report

Abstract Code

- User clicked on **View Category Report** button from Main Menu.
- Run the **View Category Report**: query for information about products and manufacturer condition for each category.
- Find the **category** using **category**[name]
- For each **category** – **product** relationship (**Label** relationship), find the category name, manufacturer name from **Manufacturer** table, retail price from **Product** table
- Count product as number of products, count distinct(manufacturer) as number of unique manufacturers, get average of retail price as average retail price for each category for records in **Label** relationship
- Display the category name, number of products, number of unique manufacturers, average retail price for each category; sorted by category name ascending.
- When **Main Menu** button is pushed:
 - Go to the Main Menu form.
- When **Log out** button is pushed:
 - Go to the Login form.

View Actual versus Predicted Revenue for GPS units Report

Task Decomposition

Lock Type: Read only on **Manufacturer, Product, Store, Category, Date** tables

Number of Locks: Multiple

Enabling Conditions: user successfully login and clicked **View Actual versus Predicted Revenue for GPS units Report**

Frequency: Low. Depends on the business requirement.

Consistency (ACID): is not critical.

Subtasks: Mother task is not needed.

View Actual versus
Predicted Revenue for
GPS units Report

Abstract Code

- User clicked on **View Actual versus Predicted Revenue for GPS units Report** button from Main Menu.
- Run the **View Actual versus Predicted Revenue for GPS units Report**: query for information about revenue and predicted revenue condition for each product in specific date range.
- User enters volume increased by percentage after discount ('increased percentage') input field.

Project Phase 1 | CS 6400 – Spring 2019 | Team 024

- When **Save** button is pushed:
 - If 'increased percentage' is blank:
 - Set [increased percentage] = 25% with message, 'default 25%'.
 - Else:
 - Set [increased percentage] = 'increased percentage' with success message
- Find all the products under GPS category from **category – product** relationship (**Label** relationship) as the target products
- Filter the products by the target products in **SaleRecords** relationship as target sale records
- Find the sales date, product name, quantity, and retail price from **Product** table for each target sale record
- Find the sale price from **onSale** relationship for each target sale record. If there is match record in **onSale** relationship with the product ID (**Product** [PID]) and sale date from the **SaleRecords** relationship, price would be the sale price which comes from the **onSale** relationship; otherwise, price would be retail price.
- Calculate the actual revenue = sales price * quantity.
- If sales price == **product** [retail price]:
 - predicted revenue = actual revenue
 - predicted quantity = quantity
- Else:
 - predicted quantity = quantity * (1 - [increased percentage])
 - predicted revenue = **product** [retail price] * predicted quantity
- Display the product ID, product name, total number of units ever sold, total number of units sold at a discount, total actual revenue, total predicted revenue, the difference between the actual revenue and predicted revenue for each product; Only predicted revenue differences $\geq 5,000$ or $\leq -5,000$ will be displayed; Sort by revenue differences in descending order.
- When **Main Menu** button is pushed:
 - Go to the **Main Menu** form.
- When **Log out** button is pushed:
 - Go to the **Login** form.

View Store Revenue by Year by State

View Store Revenue
by Year by State

Task Decomposition

Lock Type: Read only on [Store](#), [City](#), [Date](#), [OnSale](#) tables, and [SaleRecords](#) relationship.

Number of Locks: Multiple

Enabling Conditions: user successfully login and clicked *View Store Revenue by Year by State*

Frequency: Low. Depends on the business requirement.


Consistency (ACID): is not critical.

Subtasks: Mother task is not needed.

Abstract Code

- User clicks on *View Store Revenue by Year by State* button from **Main Menu**.
- When the user selects a specific state (target state) from the state drop-down box, perform the following query
 - Select the cities from the target state as target cities in the [City](#) table
 - Select the stores from the target cities as target stores in the [Store](#) table
 - Keep the sale records in the target stores as the target records from the [SaleRecords](#) relationship
 - Find the year from the [Date](#) table for each record for the target records by matching [date](#)
 - Find the retail price from the [Product](#) table for each record in [SaleRecords](#) relationship using PID
 - Find the sale price from the [OnSale](#) relationship for each record in [SaleRecords](#) relationship by matching PID and date
 - Calculate the revenue for each record in [SaleRecords](#) relationship:
If sale price is null: revenue = retail price × quantity
otherwise: revenue = sale price × quantity
 - Sum up the revenue for each target store by year
 - Find the store ID, store address from the [Store](#) table using store number
 - Find the city name from the [City](#) table
- Display the store ID, store address, city name, sales year and total revenue for each target store; sort by year in ascending order and then by revenue in descending order
- When *Main Menu* button is pushed:
 - Go to the **Main Menu** form.
- When *Log out* button is pushed:
 - Go to the **Login** form.

View Air Conditioners on Groudhog Day



View Air Conditioners
on Groudhog Day

Task Decomposition

Lock Type: Read only on [Product](#), [Category](#), [Date](#) tables, and [SaleRecords](#) relationship.

Number of Locks: Multiple

Enabling Conditions: user successfully login and clicked *View Air Conditioners on Groudhog Day*

Frequency: Low. Depends on the business requirement.

Consistency (ACID): is not critical.

Subtasks: Mother task is not needed.

Abstract Code

- User clicks on *View Air Conditioners On Groudhog Day* button from **Main Menu**.
- Perform the following query
 - Select the products (PID) that belong to the category of air conditioners from the [Product](#) table as the target products
 - Select the records for the target products in [SaleRecords](#) relationship as target records
 - Find the holiday name, and year from the [Date](#) table for the target records
 - Sum up the quantities of the target records for each year as the total number of items sold of that year
 - Divide the sum of quantities by 365 for each year as the average number of units sold per day of that year
 - Sum up the quantities of the target records on Groundhog Day for each year as the total number of units sold on Groundhog Day of that year
- Display the total number of items sold that year, the average number of units sold per day, and the total number of units sold on Groundhog Day of that year for each year; sort by year in ascending order.
- When **Main Menu** button is pushed:
 - Go to the **Main Menu** form.
- When **Log out** button is pushed:
 - Go to the **Login** form.

View State with Highest Volume for each Category

View State with
Highest Volume for
each Category

Task Decomposition

Lock Type: Read only on [Product](#), [City](#), [Store](#), [Date](#) tables, and [SaleRecords](#) relationship.

Number of Locks: Multiple

Enabling Conditions: user successfully login and clicked *View State with Highest Volume for each Category*

Frequency: Low. Business requirement to view this monthly.

Consistency (ACID): is not critical.

Subtasks: Mother task is not needed.

Abstract Code

- User clicks on *View State with Highest Volume for Each Category* button from **Main Menu**.
- User select the year (target year) and month (target month)
- Perform the following query
 - Find the year and month from the [Date](#) table for each record in [SaleRecords](#) relationship by matching date
 - Select the records in the target month and target year from the [SaleRecords](#) relationship as the target records
 - Find the category from the [Product](#) table for each record in target records using PID
 - Find the state of the store from the [City](#) table for each record in the target records using store number
 - Sum up the quantities of the unit by each category and each state as the total number of units
 - Find the state with the highest total number of units for each category
- Display the category name, state that sold the highest number of units in that category, and the number of unites that were sold in the states; sort by the category name ascending. Each category will have a **DETAILS** button.
- If user clicks on **DETAILS** button for a specific category (target category), the state with the highest number of units sold in that category in that year and month would be the target state; perform the following query
 - Find the year and month from the [Date](#) table for each record in [SaleRecords](#) relationship

Project Phase 1 | CS 6400 – Spring 2019 | Team 024

- Select the records in the target month and target year from the [SaleRecords](#) relationship as the target records
 - Find the category from the [Product](#) table for each record in target records.
 - Filter the records in the target records by the category to the target category as the target records
 - Find the city and state of the store from the [City](#) table for each record in the target records
 - Filter the records in the target records by the state to the target state as the target records
 - Select the distinct stores from the target records as the target stores
 - Find the store ID, address and cities from the [Store](#) Table of the target stores
 - Find the manager names and email address for each store
- Display the target category, target year, target month, target state as the summary. Note, the stores here would only include the stores that have sold any unit in the target category for the specific year and month.
 - Also display the store ID, address, city, manger name and email address for each manager in each target store; order by store ID ascending.
 - When **Main Menu** button is pushed:
 - Go to the **Main Menu** form.
 - When **Log out** button is pushed:
 - Go to the **Login** form.

View Revenue by Population



View Revenue by
Population

Task Decomposition

Lock Type: Read only on [Product](#), [City](#), [Store](#), [Date](#) tables, and [SaleRecords](#) relationship.

Number of Locks: Multiple

Enabling Conditions: user successfully login and clicked **View Revenue by Population**

Frequency: Low. Depends on the business requirement.

Consistency (ACID): is not critical.

Subtasks: Mother task is not needed.

Abstract Code

- User clicks on **View Revenue by Population** button from **Main Menu**.
- Perform the following query

Table of Contents

Revised 02/10/2019

Project Phase 1 | CS 6400 – Spring 2019 | Team 024

- Find the year from the Date table for each record in [SaleRecords](#) relationship by matching on date
 - Find the city of the store from the [Store](#) table for each record in [SaleRecords](#) relationship using store number
 - Find the retail price from the [Product](#) table for each record in [SaleRecords](#) relationship using PID
 - Find the sale price from the [OnSale](#) relationship for each record in [SaleRecords](#) relationship using PID and date information
 - Calculate the revenue for each record in [SaleRecords](#) relationship:
If sale price is null: revenue = retail price × quantity
otherwise: revenue = sale price × quantity
 - Sum the revenue for each city and each year in the [SaleRecords](#) relationships as the total revenue for city by year
 - Find the population from [City](#) table for each city and categorize the population as below: 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).
 - Get the average of the total revenue for one city for each city category and each year.
- Display the average revenue by city category in ascending order and year in ascending order
 - If user selects city as row then display city in row and year in column
 - If user selects year as row then display city in column and year in row
 - When **Main Menu** button is pushed:
 - Go to the **Main Menu** form.
 - When **Log out** button is pushed:
 - Go to the **Login** form.