Team 104 Phase 1 Report

CS6400 Spring 2021

Abhik Choudhury <achoudhury64@gatech.edu>

Joseph Tadros <jtadros3@gatech.edu>

Ritu Raj <rraj35@gatech.edu>

Shashvat Sinha <ssinha96@gatech.edu>

Table of Contents

[Data Types 2](#_Toc65087854)

[Business Logic Constraints 3](#_Toc65087855)

[Task Decomposition and Abstract Code 4](#_Toc65087856)

[Main Menu: 4](#_Toc65087857)

[Update Populations 4](#_Toc65087858)

[Update Holidays: 5](#_Toc65087859)

[Report 1: 5](#_Toc65087860)

[Report 2 6](#_Toc65087861)

[Report 3 7](#_Toc65087862)

[Report 4 8](#_Toc65087863)

[Report 5 9](#_Toc65087864)

[Report 6 9](#_Toc65087865)

[Report 7 11](#_Toc65087866)

[Report 8 12](#_Toc65087867)

[Report 9 13](#_Toc65087868)

# Data Types

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entity | Attribute | Data Type | Description | Example |
| Store | **Store Number** | Integer | 10-digit integer | 1324605879 |
| Store | **Phone Number** | Varchar | 10-digit integer | 5162243274 |
| Store | **Street Address** | Varchar | Max 60-digit string | 123 abc rd |
| Store | **Childcare Limit** | Integer | Max 3-digit integer to indicate minutes | 60 |
| Store | **Snack Bar** | Boolean | (Y/N) or (True/False) | Y or True |
| Store | **Restaurant** | Boolean | (Y/N) or (True/False) | Y or True |
| City | **Name** | Varchar | Max 30-character string | New York City |
| City | **State** | Varchar | 2-character string | NY |
| City | **Population** | Integer | Max 15-digit integer | 1528235 |
| Sales | **Qty** | Integer | Max 15-digit integer | 1420 |
| Product | **PID** | Integer | Max 13-digit integer | 1902348675 |
| Product | **Name** | Varchar | Max 30-character string | Super Comfy Chair |
| Product | **Retail Price** | Decimal | Max 15-digit decimal with 2 decimal places | 132.40 |
| Category | **Name** | Varchar | Max 30-character string | Sofas |
| Discount | **Discount Price** | Decimal | Max 15-digit Decimal with 2 decimal places | 98.23 |
| Date | **Date** | Date | YYYY-MM-DD | 2021-02-15 |
| Campaign | **Description** | Varchar | Max 60-character string | Summer Radio Ads |
| Holiday | **Holiday Name** | Varchar | Max 300-character string | Christmas |

We have used foundational data types. However, some data types may change in later phases of the project depending on our choice of database. For example, MySQL does not have a Boolean data type whereas PostgreSQL does.

# Business Logic Constraints

1. A **Store’s** **Childcare Limit** must not be below 0
2. A **Store’s** **Store Nbr**, **Phone** **Number**, **Street** **Address**, **Childcare** **Limit**, **Snackbar**, **Restaurant** cannot be Null
3. **City** **Name** must be in United States
4. **City** **State** must be in United States
5. **City** **Name**, **State**, and **Population** cannot be Null
6. **Product** cannot be sold at **Retail** **Price** and **Discount** **Price** for a single date
7. The **Discount** **Price** that is related to a **Product** must be less than the **Retail** **Price** of the **Product**
8. **Product** **PID**, **Name**, and **Retail** **Price** cannot be Null
9. Standard UPC/EAN codes should be used as **PID**
10. **Discount Price** and **Retail** **Price** should be greater than 0

# Task Decomposition and Abstract Code

## Main Menu:

### Task Decomposition

**Lock Types**: Read only

**Number of Locks**: 1

**Enabling Conditions**: Accessing the URL

**Frequency**: NA

**Consistency (ACID)**: Not critical

**Subtasks**: NA

### Abstract Code

Display Count of **Store Nbr** from **Store**

Display Count of **Store Nbr** from **Store** where **Restaurant** = Y or **Snack bar** = Y.

Display Count of **Store Nbr** from **Store** where **Childcare Limit** greater than 0.

Display Count of **PID** from **Product**

Display Count of **Description** from **Campaign**.

## Update Populations

### Task Decomposition

### 

**Lock Types**: One write-only on **City Population**

**Number of Locks**: 1

**Enabling Conditions**: Click on “Update Population”

**Frequency**: NA

**Consistency (ACID)**: Until write operation finishes, no other user should be able to view **City Population**

**Subtasks**: NA

### Abstract Code

Given the **Name**, **State** find the **Population** of **City**

Update **Population**

## View/Update Holidays:

### Task Decomposition

**Lock Types**: Ready-only & Write-only lock on **Holiday Holiday Name**

**Number of Locks**: 2

**Enabling Conditions**: Click on “View/Update Holiday” button enables “View Holiday Information” subtask. User submitting **Holiday Name** and **Date** enables “Update/Edit Holiday” subtask.

**Frequency**: Each subtask will have different frequency

**Consistency (ACID)**: Until write operation finishes, no other user should be able to view **Holiday**

**Subtasks**: All tasks must be done. “View Holiday Information” occurs first, followed by “Update/Edit Information”**.** Mother Task is needed.

### Abstract Code

If user clicks “View/Update Holiday”:

Display list of **Holiday Name** and **Date**

If user clicks “Update/Edit Holiday” and provides Date and Holiday Name:

Add/update new **Holiday Name** and **Date**

## Report 1:

### Task Decomposition

**Lock Types**: 2 Read-only lock on **Product** and **Category**

**Number of Locks**: 2

**Enabling Conditions**: Click on “Category Report”

**Frequency**: All 2 have the same frequency

**Consistency (ACID)**: NA

**Subtasks**: All tasks must be done. “Find Category” occurs first, followed by second sub-task**.** Mother Task is needed.

### Abstract Code

Get List of Categories sorted alphabetically

For Each **Category**

Get all **Products** and **Retail Price** that match the category

Count all **Products**

Average their **Retail Price**

Find the Max **Retail Price**

Find the Min **Retail Price**

Display the four numbers

## Report 2

### Task Decomposition

**Lock Types**: 4 Read-only lock on **Product, Sales, Dates** and **Discount**

**Number of Locks**: 4

**Enabling Conditions**: Click on “Actual versus Predicted Revenue”

**Frequency**: All 4 have the same frequency

**Consistency (ACID)**: NA

**Subtasks**: All tasks must be done in a sequential order from left to right as shown in TD diagram. Mother task is needed.

### Abstract Code

Get a list of **PID** in category “Couches and Sofas”

For each **PID**:

Get the list of discounted **Dates**

For each discounted **Date**:

Get the **Sales Qty** for this PID for this date

Get the discounted **Price**, **Retail Price** for this **PID** and **Date**

Calculate Actual Revenue as discounted **Price** \* **Sales Qty**

Calculated Projected Qty as **Sales Qty**\*0.75

Calculate Projected Revenue as **Retail Price** \* ProjectedQty

Sum up the Actual Revenue, Projected Revenue, Sales Qty, Projected Qty

Calculate Revenue Difference as Actual Revenue – Projected Revenue

Display **PID**, **Product Name**, **Retail Price**, **Sales Qty,** Actual Revenue, Predicted Qty, Projected Revenue

Order by Diff Desc

## Report 3

### Task Decomposition

**Lock Types**: 4 Read-only lock on **Store, Sales, Dates** and **City**

**Number of Locks**: 4

**Enabling Conditions**: Click on “Store Revenue by Year by State”

**Frequency**: All 4 have the same frequency

**Consistency (ACID)**: NA

**Subtasks**: All tasks must be done in a sequential order from left to right as shown in TD diagram. Mother task is needed.

### Abstract Code

Get list of **States** for all **Cities**

Display **States**

When a button is pushed, then do the following:

Get list of Stores using the selected **State**

For each **Store:**

Get **Store** **Nbr** and **Store** **Address**

Get **City** **Name** from **City**

Get list of **Sales** for **Store**

Get list of **Dates** from **Sales**

Calculate list of Years from **Date**

For each Year:

For each **Sale** within Year:

Get **Date** of **Sale**

Get **PID** of **Sale**

If **Discount** of **Date** and **PID** exists:

Get **Discount** **Price** as Price

Else:

Get **Retail** **price** as Price

Get **Sales** **Qty** of **Sale**

Sum (**Sales** **Qty** \* Price) for each Year AS Total Year Revenue

Sum Total Year Revenue for each **Store** AS Total Revenue

Display **Store** **Nbr**, **Store** **Address**, **City** **Name**, Year, Total Revenue

Sort by Total Revenue descending and Year ascending;

## Report 4

### Task Decomposition

**Lock Types**: 4 Read-only lock on **Sales, Dates, Product** and **Category**

**Number of Locks**: 4

**Enabling Conditions**: Click on “Outdoor Furniture on Groundhog Day”

**Frequency**: All 4 have the same frequency

**Consistency (ACID)**: NA

**Subtasks**: All tasks must be done in a sequential order from left to right as shown in TD diagram. Mother task is needed.

### Abstract Code

Get list of Years

For each *Year*:

Display *Year*

Find start Date and end **Date** of year from stored **Dates**

Find the **Quantity**, **Product** **PID**, **Category Name** stored in **Sales** for the date range

Filter the Sales data based on outdoor furniture product **Category**

Calculate *Total Units Sold in Year* = Sum of **Sales** **Quantity** for all **PID** for each **Date**

Display *Total Units Sold in Year*

Calculate *Avg. Units Sold per Day* = Total Sales/365

Display *Avg. Units Sold per Day*

Get Feb 2nd as **Date** store in **Dates** for the Year

Change start Date and end Date to Feb 2nd

Find the **Quantity**, **Product** **PID**, **Category** **Name** stored in **Sales** for the date range

Filter the **Sales** data based on outdoor furniture product **Category Name**

Calculate *Units Sold on Ground Hog Day* = Sum of **Sales Quantity** for all PID for the **Date**

Display *Units Sold on Ground Hog Day*

After iterating for all listed years:

Sort the output display by ascending Order of *Year*

## Report 5

### Task Decomposition

**Lock Types**: 6 Read-only lock on **Dates, Product, Sales, Store, City** and **Category**

**Number of Locks**: 6

**Enabling Conditions**: Click on “State with Highest Volume for each Category”

**Frequency**: All 6 have the same frequency

**Consistency (ACID)**: NA

**Subtasks**: All tasks must be done in a sequential order from left to right as shown in TD diagram. Mother task is needed.

### Abstract Code

Get list **Date** from **Dates**

Calculate list of Months and Years from **Date**

Display the year and month

Let the user select one year-month set.

Get the list of Categories in ascending order.

With each Category

Get the list of **PID** for that **Category**

With each **PID**

Get the Sales for that **PID** for the required Year and Month.

For each **Sale**

Get the **Store Number** for the sale

Get the **State** for that **Store Number**

Sum up all the Sales quantities by **State**

Select the **State** (or **states**) with the Highest Total Unit Sale

List out the top State or **States**

## Report 6

### Task Decomposition

**Lock Types**: 5 Read-only lock on **Dates,** **Product, Sales, Store, City** and **Discount**

**Number of Locks**: 5

**Enabling Conditions**: Click on “Revenue by Population”

**Frequency**: All 5 have the same frequency

**Consistency (ACID)**: NA

**Subtasks**: All tasks must be done in a sequential order from left to right as shown in TD diagram. Mother task is needed.

### Abstract Code

Get list of all **City Name** and their **Population**

For each **City Name:**  
Assign Each city to a population category using condition\*\*

For each population category:

Get the list of cities for that category

For each city**:**

Get the list of stores in the city

For each **Store:**

Get the total sales by **PID** by **Date**  
For each **PID**:

If the **Sales** happed on a **Discount** **Date**:

Then Revenue is **Sales**\* **Discount price**

Else:

Revenue is **Sales**\* **Retail price**

Sum up revenue by year

Sum up annual revenues for this category

*\*\* 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).*

## Report 7

### Task Decomposition

**Lock Types:**

**Number of Locks:**

**Enabling Conditions:**

**Frequency:**

**Consistency (ACID):**

**Subtasks:**

### Abstract Code

Get list of **Date** of **Sales**

Calculate list of Months from **Date**

Limit Month list to most recent 12 months

Get list of **Childcare** **Limits**

For each **Childcare** **Limit**:

Get list of **Store Nbr** with **Childcare** **Limit**

For each Month:

For each **Store Nbr**:

Get list of **Sales**

For each **Sale:**

Get **Date** of **Sale**

Get **PID** with **Sale**

If **Discount** of **Date** and **PID** exists:

Get **Discount** **Price** as Price

Else:

Get **Retail** **price** as Price

Get **Sales** **Qty** of **Sales**

Sum (**Sales** **Qty** \* Price) as Total Revenue

Display **Childcare** **Limit**, Month, Total Revenue

IF **Childcare** **Limit** = 0:

**Childcare** **Limit** = “No Childcare”

## Report 8

### Task Decomposition

**Lock Types:**

**Number of Locks:**

**Enabling Conditions:**

**Frequency:**

**Consistency (ACID):**

**Subtasks:**

### Abstract Code

Get **Quantity**, **Product** **PID**, **Category Name, Store Restaurant** stored in **Sales**

Filter out the sales data where **PID** is not present

Combine the data set based on **Category Name**

For every **Category Name**:

Display **Name**

Classify each **Store** within **Category** based on **Restaurant** value

Filter **Restaurant** = Y

Display Store Type *= Restaurant*

Sum all **Quantity** sold across **Store**

Display as *Quantity Sold*

Filter **Restaurant** = N

Display Store Type *= Non-restaurant*

Sum all **Quantity** sold across **Store**

Display as *Quantity Sold*

Repeat the step until all Categories completed

Arrange the report with ascending order of **Category Name**

## Report 9

### Task Decomposition

**Lock Types:**

**Number of Locks:**

**Enabling Conditions:**

**Frequency:**

**Consistency (ACID):**

**Subtasks:**

### Abstract Code

Get the list of all **PID**

For each **PID**

Get the **Discount Date** for the **PID**

For each **Discount Date**

Get the **Sales** for that **PID** for that **Date**

Check to see if the **Discount Date** also had a **Campaign** running on the same **Date**.

Assign the **Sales** to either the Campaign or non-Campaign bucket.

Sum up the **sales** for each bucket.

Get the differences for each bucket

Put all the **PID, Product Names**, campaign, non-campaign and difference numbers in descending order by difference.

Display the top 10 and bottom 10.