**Store Data Table Generation :**This table includes data related to the GameStop Stores.

Store Number Starts from 1010 and is sequentially incremented by 10 until it reaches the 2000  
Address was generated from the following website: <https://www.randomlists.com/random-addresses>  
Store Opening and Closing hours are defaulted to open from 10 AM until 7 PM for all stores.

**Zip\_tax Data Table Generation:**This table includes data related to the tax rates for the GameStop Stores.

Zipcode tax rates used the following script to generate random tax values between 3.5% and 14.5% where the zipcodes are pulled from the Store Data table

public static void taxRate(int val){  
 double rate;  
 for (int y = 0; y<val; y++) {  
 Random rand = new Random();  
 rate = 3.5 + rand.nextDouble()\*11;  
 System.*out*.println(rate);  
 }  
}

**Taxed Data Table Generation:**This table contains data for the tax rate for the stores and includes the corresponding store ID linked to the zipcode.

TaxID is a unique number associated with each store starting from 10001 and increasing sequentially. The Store ID also includes the ZIP code of the store address. These were pulled from the Store data and manually generated.

**Product Data Table Generation:**This table contains the inventory of products for sale on the gamestop website.

Product IDs were manually assigned starting from 10001001 and assigning random values according to the type of product. Numbering of these products were somewhat sequential but varied depending on the procut. I.e. All Xbox console product codes range from 10001001 to 10001010. Playstation product codes range from 10002001 to 10002010 and so on and so forth.  
Product Names were taken from the GameStop website and entered manually  
Product Type was assigned a value of Console&Hardware, VideoGames, or Board Games  
Condition was either New or Pre-owned and the prices were pulled from the GameStop website as well.  
Warnings were generated randomly by assigning warning labels based upon basic knowledge of the video games.

**Publisher Data table:**This table contains the data for the publishers of the product data table

This table includes a publisher\_ID and a publisher name. This is taken directly from the product data table and duplicates are removed. This was a manual process. Publisher IDs were also manually assigned and generated

**Publishes table:**This data table contains the publisher ID and the product codes for each individual product.

Publishes\_ID table includes a unique key for the publishing relationship and also includes the publisher ID and the corresponding product code.

**Belongs\_to Data table generation:**  
This table is essentially the inventory of the all the different stores and product codes. As such, the size of this table is 100x100. The first column is the Store ID and the Second column is the Product ID. For all 100 items this is repeated. Then the price, the product count (in stock), products sold (revenue) and products returned.

To generate this data, I utilized a table with the store ID and product code. The product count is a random integer from 1-15, the products sold is a random integer from 0-6, and the products returned is a random integer from 0 to 1.

The java code to create this table is below:

import java.awt.\*;  
import java.io.\*;  
import java.util.Random;  
import java.util.Scanner;  
import javax.swing.JFileChooser;  
import java.io.File;  
  
public class db\_inventory {  
 private String[] ProdCode;  
 private String[] StoreID;  
  
  
 public static void main(String[] args) throws FileNotFoundException  
 {  
 String[] ProdCode = new String[100];  
 String[] StoreID = new String[100];  
 String[] ProdPrice = new String[100];  
 int count = 0;  
 String line = "";  
 String splitBy = ",";  
 try {  
 BufferedReader br = new BufferedReader(new FileReader("C:\\Users\\stanw\\OneDrive\\Documents\\College Classes\\05\_Fall 2021\\CSC6710\\Data\\Book2.csv"));  
  
 while ((line = br.readLine()) != null) *//returns a Boolean value* {  
 String[] Values = line.split(splitBy); *// use comma as separator  
 //System.out.println("Product Code ID: " + Values[0] + ", Store ID:" + Values[1]);* ProdCode[count] = Values[0];  
 StoreID[count] = Values[1];  
 ProdPrice[count] = Values[2];  
 count++;  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 for (int i=0;i<100;i++){  
 for (int j=0;j<100;j++){  
 Random r1 = new Random();  
 int prodCount = r1.nextInt(15)+1;  
 int prodSold = r1.nextInt(7);  
 int prodReturned = r1.nextInt(2);  
 System.*out*.println(StoreID[i]+","+ProdCode[j]+","+ProdPrice[j]+","+prodCount+","+prodSold+","+prodReturned);  
  
 }  
 }  
 }  
  
  
 public static void genEmptyTable(){  
  
 }  
}

**Customer Data Table Generation:**This table contains all of the customer information.

Customer ID starts from 10000001 and is sequentially numbered until 100 entries were populated.

Names were generated from the following website: <https://www.randomlists.com/random-names>  
Addresses were generated from the following website: <https://www.randomlists.com/random-addresses>

Phone numbers were generated from the following website: <https://www.randomlists.com/phone-numbers>

Email addresses were an aggregate of the First and Last name followed by @gmail.com with no accounting for numbers.

Date of Birth was generated through a Java script below:

public static String RandomDOB(){  
 GregorianCalendar gc = new GregorianCalendar();  
 int year = *randBetween*(1960, 2010);  
 gc.set(gc.*YEAR*, year);  
  
 int dayOfYear = *randBetween*(1, gc.getActualMaximum(gc.*DAY\_OF\_YEAR*));  
  
 gc.set(gc.*DAY\_OF\_YEAR*, dayOfYear);  
  
 return gc.get(gc.*MONTH*) + "-" + (gc.get(gc.*DAY\_OF\_MONTH*) + 1) + "-" + gc.get(gc.*YEAR*);  
}  
public static int randBetween(int start, int end) {  
 return start + (int)Math.*round*(Math.*random*() \* (end - start));  
}

public static void DOBGen(){  
 for(int i=0;i<100;i++){  
 System.*out*.println(*RandomDOB*());  
 }  
}

Credit card numbers were generated with the script below:

public static void ccNumGen(){  
 for (int y = 0; y<100; y++) {  
 for (int i = 0; i < 16; i++) {  
 Random rand = new Random();  
 System.*out*.print(rand.nextInt(10));  
 }  
 System.*out*.println();  
 }  
}

Credit Card Expiration was generated using the following script

public static void ccEXPGen(){  
 for (int y = 0; y<100; y++) {  
  
 Random rand = new Random();  
 System.*out*.print(rand.nextInt(12)+1); *//Random month* System.*out*.print(", "+(rand.nextInt(4)+22)); *//Expire year between 22 to 25* System.*out*.println();  
 }  
}

Credit Card Security Number was generated using the following script

public static void csvGEN(){  
 for (int y = 0; y<100; y++) {  
 for (int i = 0; i < 3; i++) {  
 Random rand = new Random();  
 System.*out*.print(rand.nextInt(10));  
 }  
 System.*out*.println();  
 }  
}

**Shopping\_Cart data table generation**This table contains the cart ID as well as the customer ID and the order date.  
  
This table was generated manually from the data that was created for the cart\_detail table by removing the duplicates in the cart\_ID table.

**Cart\_Detail data table generation**This table contains the information regarding sales of products.  
  
There is a unique cart detail ID, the store ID of the store selling the product, a cart ID for each purchase, the product ID order, the quantity ordered and purchase date.

The purchase date was assigned manually where the rest of the values were generated from the following script. It selects a random store id, a random product ID, and from 1-5 items for the customer to purchase.

See the script below.

import java.io.BufferedReader;  
import java.io.FileNotFoundException;  
import java.io.FileReader;  
import java.io.IOException;  
import java.util.Random;  
  
public class cart\_gen {  
 public static void main(String[] args) throws FileNotFoundException  
 {  
 String[] ProdCode = new String[100];  
 String[] StoreID = new String[100];  
 String[] ProdPrice = new String[100];  
 int count = 0;  
 int cartstart = 1000001;  
 int custStart= 100000001;  
 String line = "";  
 String splitBy = ",";  
 String FileName ="C:\\Users\\stanw\\OneDrive\\Documents\\College Classes\\05\_Fall 2021\\CSC6710\\Data\\Book2.csv";  
  
 try {  
 BufferedReader br = new BufferedReader(new FileReader(FileName));  
  
 while ((line = br.readLine()) != null) *//returns a Boolean value* {  
 String[] Values = line.split(splitBy); *// use comma as separator  
 //System.out.println("Product Code ID: " + Values[0] + ", Store ID:" + Values[1]);* ProdCode[count] = Values[0];  
 StoreID[count] = Values[1];  
 ProdPrice[count] = Values[2];  
 count++;  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 for (int i=0;i<100;i++){  
 int rstore,rprod;  
 Random r1 = new Random();  
 int jran = r1.nextInt(4)+1;  
 int cartID = cartstart+i;  
 int custID = custStart+r1.nextInt(100)+1;  
 rstore = r1.nextInt(99);  
 for (int j=0;j<jran;j++){  
 rprod = r1.nextInt(99);  
 int rqty = r1.nextInt(2)+1;  
 double nPrice = rqty\*Double.*valueOf*(ProdPrice[rprod]);  
 System.*out*.println(cartID+","+custID+","+StoreID[rstore]+","+ProdCode[rprod]+","+rqty+","+nPrice);  
 *//Change this line for matching data entry* }  
  
 }  
 }  
  
}