Cassie Boylan, DH Kim, Alexis Mekueko

DATA 607 | Project 3

Project 3 A

group project - Data analysis of normalized data

**Contents**

[I. Data Source 1](#_Toc53220414)

[II. Collaboration Tools 1](#_Toc53220415)

[III. How Data was Imported into Global Environment 1](#_Toc53220416)

[IV. Overview of Dataset 2](#_Toc53220417)

[V. Assumptions Made 3](#_Toc53220418)

[VI. Questions to Explore/Answer 3](#_Toc53220419)

[VI. Entity Relationship Diagram: 4](#_Toc53220420)

## I. Data Source

Dataset was sourced from the machine learning databases under the UC Irvine ML Repository.

link:

## II. Collaboration Tools

Collab tools for this project:

* Google Docs
* Microsoft Teams

## III. How Data was Imported into Global Environment

Dataset was pulled using GET() method, passing the url of the page to retrieve a “response object”

This is an http method that allows you to read all the html information (attribute tags and values) from the GET output. The response object in this case is the dataset located at the url passed.

The write-disk function was passed into GET as second argument; utilized to specify where in local machine memory to catch the response object returned by GET() and url.

We specify the <where to store >by calling tempfile() function, a useful function to create temporary files to store output (in this case our dataset from the GET method). The function tempfile() will accept a pattern argument to specify the prefix naming convention to assign temp file(s) (optional), a directory path on where to store the file (tmpdir) and the type of temporary file to create (fileext) – which will be assigned by passing the filetype suffix. We are passing the filetype “.xlsx” to specify that dataset be saved in a new temporary Excel workbook.

Excel is an ideal file format for when you have multiple dataset to import & store at once as each can be saved in a separate worksheet in the same file.

From here, we can read in a single worksheet/dataset through the use of read\_excel(). The main advantage of using read\_excel is that (like read\_csv) the data imports into our environment as an object with 3 attributes : table\_df, tbl, data.frame. The tbl\_df will enable us to easily apply dplyr and tidyr functionality that we inevitably will need.

We needed to create a temporary file to save the dataset in local memory to read in the Excel file because, unlike read\_csv which accepts data stored elsewhere, Excel spreadsheets must be stored locally to be read and cannot be read from a url.

Notice that we assigned the temp file we created a variable name so that we could reference the variable name when specifying file to read in as the first argument.

The read\_excel function will only read in one worksheet from an Excel workbook at a time; you can specify which worksheet to read in by passing the name of the worksheet as the second argument. If we left this argument blank, read\_excel would default to reading in the first worksheet.

We called read\_excel twice to read in the dataset of each worksheet that we imported and saved to Excel.

With the 2 datasets we’ve imported, saved to a temporary Excel file and read in as tbl\_df objects to access in our global environment, we would now would like to combine the 2 tbl\_df objects into one for cleaner, more encompassing dataset for dplyr analysis. To do that, we use an rbind() function. The variables in the 2 datasets are identical to each other so we used a simple rbind() function to combine the two datasets vertically ( datasets stacked on top of each other). The same variables must be located in both datasets for this type of combining by rows to make sense. If one dataset had had more variables (columns) that the other, we could have either dropped the columns not shared or used rbind.fill in the plyr package to populate NA values for the excess.

Since these 2 datasets shared the same variables and combined with a simple rbind, we will run a check for any duplicated rows that may have existed in both datasets. We now have a dataset of 8 columns and over 1 million observations.

## IV. Overview of Dataset

Our dataset stores the information on invoices generated for orders from a holiday homegoods store. Each row/observation contains the relevant variables of an invoice line item:

* SKU
* SKU description
* Invoice id
* Date invoice generated
* SKU quantity ordered
* Per unit SKU cost billed
* Customer id who purchased
* Country of Customer

We can see repeating values in dataset for invoice id and customer id as each row is a single invoice line and unsurprisingly multiple SKUs were ordered by same customer and billed on same invoice as part of order.

The repeating values gave a good indication that this dataset is likely already pretty tidy.

The dataset is restricted in its Date column to only invoices generated within the first day of December for years 2009 and 2010, as well as the 4 months post December for years 2010 and 2011. This slicing can help inform understanding holiday sales as well as reviewing demand in the 4 months post holiday.

Date values include the date as well as the time stamp invoice generated. Time stamps are recorded in European/Military time and we will need to convert timezone to US Eastern to better understand/communicate times.

## V. Assumptions Made

Assumptions we ae making in interpreting this dataset:

* That invoices are generated at time of sales order as most consumer sales are – We will infer that invoice timestamp will refer to the date/time order by customer was made.
* That **Country** variable refers to where the order originated/where the customer resides. Bill-to addresses on invoices can differ from where order was placed or where order was shipped. To simplify our analysis, we will infer that country relates to where customer who placed order resides.

## VI. Questions to Explore/Answer

Possible Questions we are Interested in Answering:

* How did revenue change between December 2009 and 2010?
* How did revenue change in the 4 months after major holiday season?
* Did unit pricing increase between 2010 and 2011? 2009 and 2010?
  + If so, by how much?
  + Were prices increased in holiday months?
  + Did prices decrease in post-holiday months?
* During what hourly time range were most orders placed?
  + Is there a preferred time window for shopping?
* What items were most popular in the holiday month vs post-holiday months?
* Did SKU popularity change between years?
* Did SKU popularity differ between countries?
* What SKUs brought in greatest revenue?
* Customers from which countries spent the most?
* Customers from which countries spent the least?
* From which countries did customers purchase the least? (SKU quantity)
* From which countries did customers purchase the most?

## VI. Entity Relationship Diagram:

This data can be stored in a normalized relational database in that variables can be broken into tables where each table represents a logical entity.

The logical entities in our data are as follows: invoice, invoice line, SKU, customer

Each variable in our dataset can be mapped back to one of these logical objects.

Together, each form an observation of an invoice line that can be grouped, summarized, and used for descriptive and inferential statical analysis.

Entity Relationship between Entities is as follows:

* Each invoice id will map to one customer id
  + A customer id can be mapped to many invoices but an invoice can be related to one customer id
  + **Invoice to Customer** 🡪 N:1
* Each invoice will map to many invoice line items
  + An invoice can contain many invoice line items but invoice line items can only map to one invoice
  + **Invoice to Invoice Line Item** 🡪 1:N
* Each invoice line will map to one SKU
  + SKUs will appear on many invoice line items but each invoice line item can only map to one SKU
  + **SKU to Invoice** 🡪 1:1