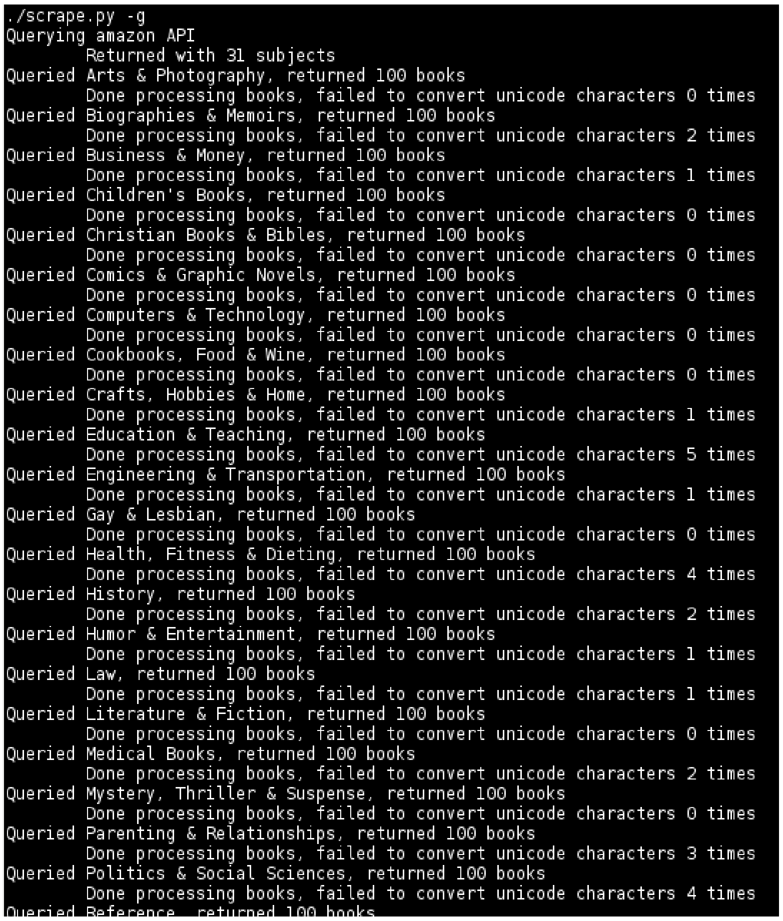
Phase Four
Book Barn Database

Team #10

Jason Boutte
Cameron Brownfield
Stacy Rios
Matthew Pastera

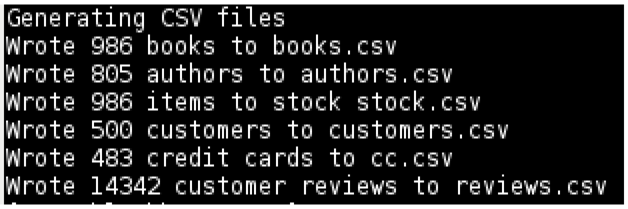
Generating Data:

We decided to gather data from two sources. Since our project is an online bookstore we chose to retrieve actual book data from Amazon and generate the remaining data from random values. To accomplish this we wrote a python script that utilizes the Amazon Product Advertising API to retrieve the book data and a python library called Faker to generate the rest. Through the Amazon API we were able to query book genres and retrieve 100 books per genre, totaling around 1000 books in all. Using the Faker library we generated realistic data to model our customers, their payment methods, book reviews and our stores stock. After all the data was gathered, our python script transformed it into csv files ready to import into our database. The script honors all our schema constraints and filters out anything that doesn't, providing customers with unique user-names, email addresses, credit cards, and addresses. The script also generates a dynamic stock for our store and the books returned from Amazon. For our project we chose to use a PostgreSQL database. To import the data we used the \copy command from the psql terminal. (**Python script at end of document**)



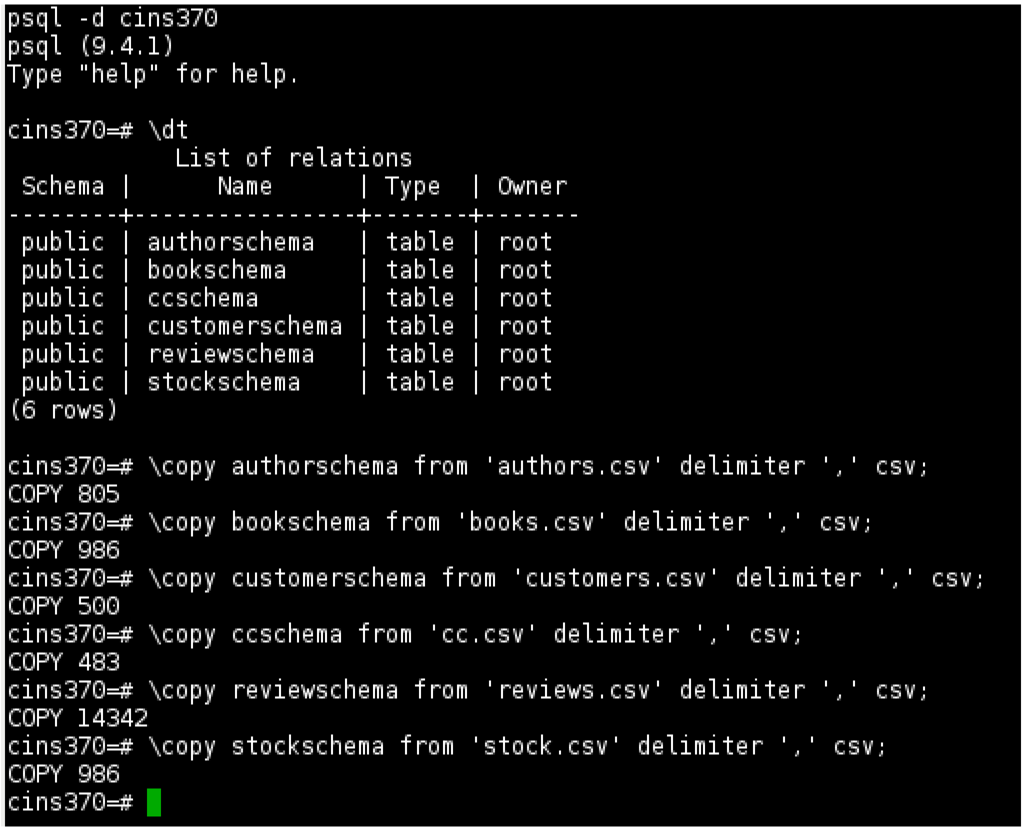
```
./scrape.py -g
Querying amazon API
        Returned with 31 subjects
Queried Arts & Photography, returned 100 books
        Done processing books, failed to convert unicode characters 0 times
Queried Biographies & Memoirs, returned 100 books
        Done processing books, failed to convert unicode characters 2 times
Queried Business & Money, returned 100 books
        Done processing books, failed to convert unicode characters 1 times
Queried Children's Books, returned 100 books
        Done processing books, failed to convert unicode characters 0 times
Queried Christian Books & Bibles, returned 100 books
        Done processing books, failed to convert unicode characters 0 times
Queried Comics & Graphic Novels, returned 100 books
        Done processing books, failed to convert unicode characters 0 times
Queried Computers & Technology, returned 100 books
        Done processing books, failed to convert unicode characters 0 times
Queried Cookbooks, Food & Wine, returned 100 books
        Done processing books, failed to convert unicode characters 0 times
Queried Crafts, Hobbies & Home, returned 100 books
        Done processing books, failed to convert unicode characters 1 times
Queried Education & Teaching, returned 100 books
        Done processing books, failed to convert unicode characters 5 times
Queried Engineering & Transportation, returned 100 books
        Done processing books, failed to convert unicode characters 1 times
Queried Gay & Lesbian, returned 100 books
        Done processing books, failed to convert unicode characters 0 times
Queried Health, Fitness & Dieting, returned 100 books
        Done processing books, failed to convert unicode characters 4 times
Queried History, returned 100 books
        Done processing books, failed to convert unicode characters 2 times
Queried Humor & Entertainment, returned 100 books
        Done processing books, failed to convert unicode characters 1 times
Queried Law, returned 100 books
        Done processing books, failed to convert unicode characters 1 times
Queried Literature & Fiction, returned 100 books
        Done processing books, failed to convert unicode characters 0 times
Queried Medical Books, returned 100 books
        Done processing books, failed to convert unicode characters 2 times
Queried Mystery, Thriller & Suspense, returned 100 books
        Done processing books, failed to convert unicode characters 0 times
Queried Parenting & Relationships, returned 100 books
        Done processing books, failed to convert unicode characters 3 times
Queried Politics & Social Sciences, returned 100 books
        Done processing books, failed to convert unicode characters 4 times
Queried Reference, returned 100 books
```

Querying Amazon



```
Generating CSV files
Wrote 986 books to books.csv
Wrote 805 authors to authors.csv
Wrote 986 items to stock stock.csv
Wrote 500 customers to customers.csv
Wrote 483 credit cards to cc.csv
Wrote 14342 customer reviews to reviews.csv
```

Transforming data



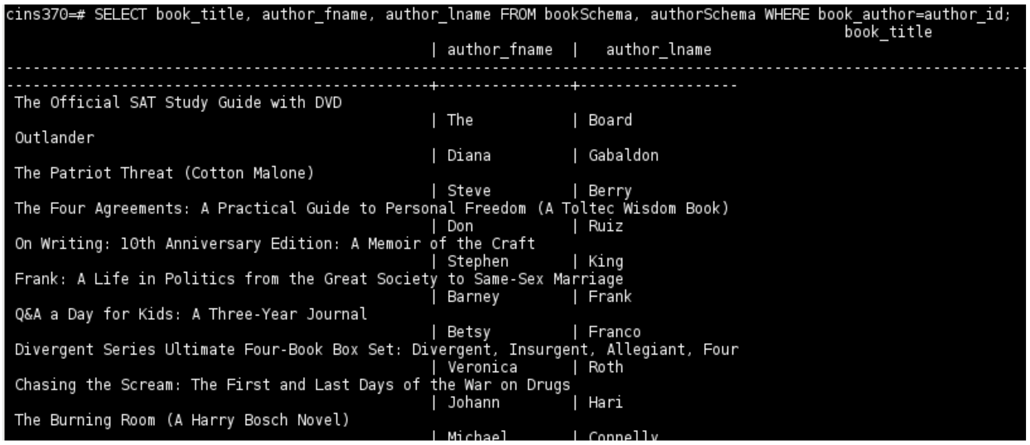
```
psql -d cins370
psql (9.4.1)
Type "help" for help.

cins370=# \dt
            List of relations
 Schema |      Name      | Type  | Owner
--------+----------------+-------+-------
 public | authorschema   | table | root
 public | bookschema     | table | root
 public | ccschema       | table | root
 public | customerschema | table | root
 public | reviewschema   | table | root
 public | stockschema    | table | root
(6 rows)

cins370=# \copy authorschema from 'authors.csv' delimiter ',' csv;
COPY 805
cins370=# \copy bookschema from 'books.csv' delimiter ',' csv;
COPY 986
cins370=# \copy customerschema from 'customers.csv' delimiter ',' csv;
COPY 500
cins370=# \copy ccschema from 'cc.csv' delimiter ',' csv;
COPY 483
cins370=# \copy reviewschema from 'reviews.csv' delimiter ',' csv;
COPY 14342
cins370=# \copy stockschema from 'stock.csv' delimiter ',' csv;
COPY 986
cins370=#
```

Importing resulting csv files.

SQL Code:

```sql
SELECT book_title, author_fname, author_lname
FROM bookSchema, authorSchema
WHERE book_author=author_id;
```

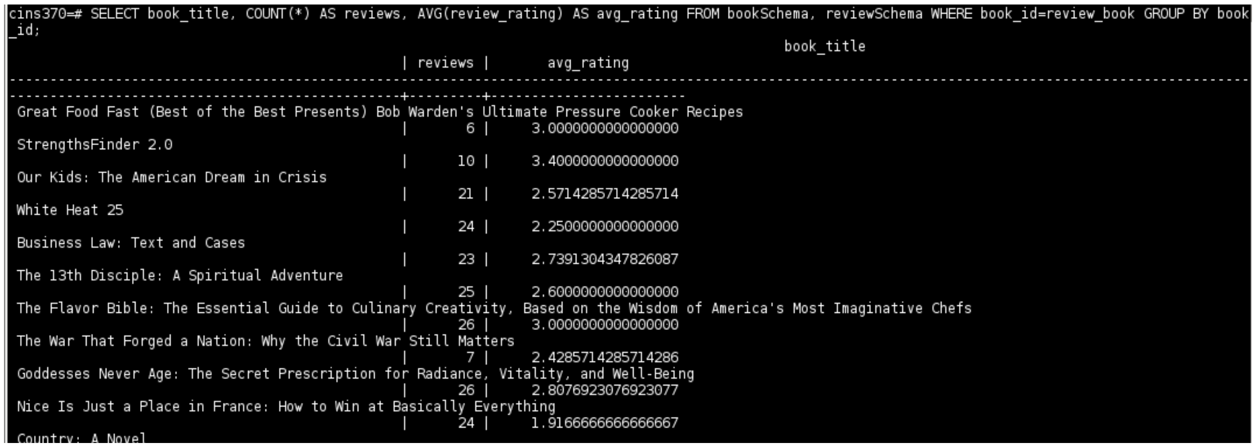
This query returns tuples for all books containing the book title and the authors name. This query can easily be modified to filter books. The shown results are truncated.



```
cins370=# SELECT book_title, author_fname, author_lname FROM bookSchema, authorSchema WHERE book_author=author_id;
                                                                                                      book_title
                                                  | author_fname  |   author_lname
------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------+---------------+-----------------
 The Official SAT Study Guide with DVD
                                                  | The           | Board
 Outlander
                                                  | Diana         | Gabaldon
 The Patriot Threat (Cotton Malone)
                                                  | Steve         | Berry
 The Four Agreements: A Practical Guide to Personal Freedom (A Toltec Wisdom Book)
                                                  | Don           | Ruiz
 On Writing: 10th Anniversary Edition: A Memoir of the Craft
                                                  | Stephen       | King
 Frank: A Life in Politics from the Great Society to Same-Sex Marriage
                                                  | Barney        | Frank
 Q&A a Day for Kids: A Three-Year Journal
                                                  | Betsy         | Franco
 Divergent Series Ultimate Four-Book Box Set: Divergent, Insurgent, Allegiant, Four
                                                  | Veronica      | Roth
 Chasing the Scream: The First and Last Days of the War on Drugs
                                                  | Johann        | Hari
 The Burning Room (A Harry Bosch Novel)
                                                  | Michael       | Connelly
```

```
SELECT book_title, COUNT(*) AS reviews, AVG(review_rating) AS avg_rating
FROM bookSchema, reviewSchema
WHERE book_id=review_book
GROUP BY book_id;
```

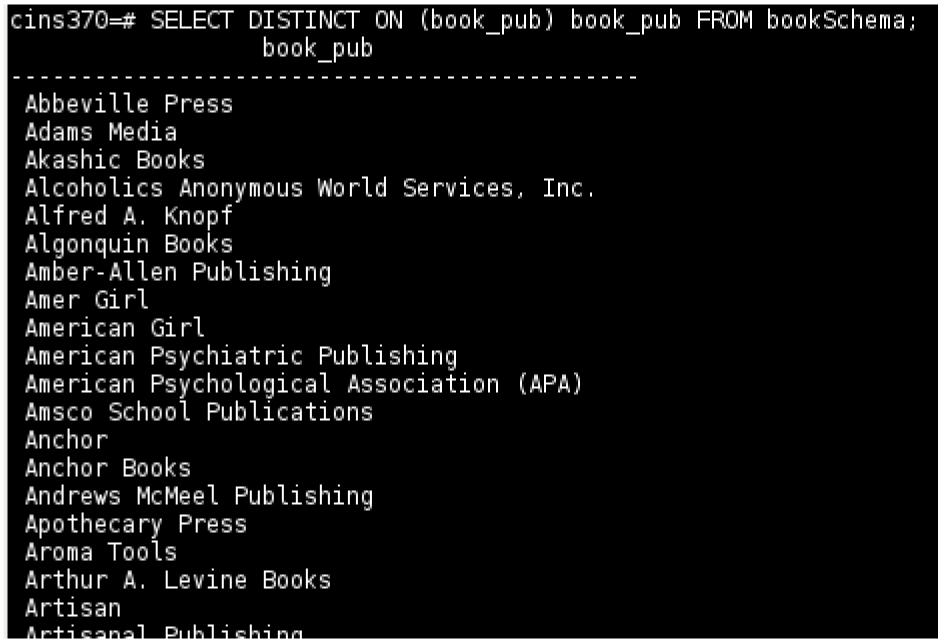
This query returns each book with the number of reviews and the average rating. The shown results are truncated.



```
cins370=# SELECT book_title, COUNT(*) AS reviews, AVG(review_rating) AS avg_rating FROM bookSchema, reviewSchema WHERE book_id=review_book GROUP BY book
_id;
                                                                                                    book_title
                                                | reviews |       avg_rating
--------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------+---------+-----------------------
 Great Food Fast (Best of the Best Presents) Bob Warden's Ultimate Pressure Cooker Recipes
                                                |       6 |     3.0000000000000000
 StrengthsFinder 2.0
                                                |      10 |     3.4000000000000000
 Our Kids: The American Dream in Crisis
                                                |      21 |     2.5714285714285714
 White Heat 25
                                                |      24 |     2.2500000000000000
 Business Law: Text and Cases
                                                |      23 |     2.7391304347826087
 The 13th Disciple: A Spiritual Adventure
                                                |      25 |     2.6000000000000000
 The Flavor Bible: The Essential Guide to Culinary Creativity, Based on the Wisdom of America's Most Imaginative Chefs
                                                |      26 |     3.0000000000000000
 The War That Forged a Nation: Why the Civil War Still Matters
                                                |       7 |     2.4285714285714286
 Goddesses Never Age: The Secret Prescription for Radiance, Vitality, and Well-Being
                                                |      26 |     2.8076923076923077
 Nice Is Just a Place in France: How to Win at Basically Everything
                                                |      24 |     1.9166666666666667
 Country: A Novel
```

```
SELECT DISTINCT ON (book_pub) book_pub
FROM bookSchema;
```

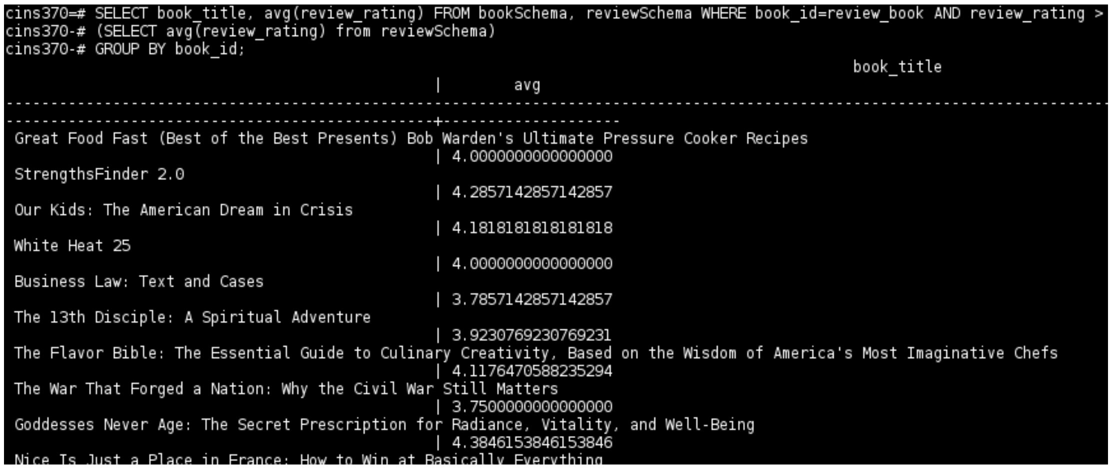
This query returns all of the unique publishers. The shown results are truncated.



```
cins370=# SELECT DISTINCT ON (book_pub) book_pub FROM bookSchema;
                  book_pub
---------------------------------------------
 Abbeville Press
 Adams Media
 Akashic Books
 Alcoholics Anonymous World Services, Inc.
 Alfred A. Knopf
 Algonquin Books
 Amber-Allen Publishing
 Amer Girl
 American Girl
 American Psychiatric Publishing
 American Psychological Association (APA)
 Amsco School Publications
 Anchor
 Anchor Books
 Andrews McMeel Publishing
 Apothecary Press
 Aroma Tools
 Arthur A. Levine Books
 Artisan
 Artisanal Publishing
```

```sql
SELECT book_title, avg(review_rating)
FROM bookSchema, reviewSchema
WHERE book_id=review_book AND review_rating > (
       SELECT avg(review_rating)
       FROM reviewSchema
       )
GROUP BY book_id;
```

This query returns only books whose average rating is greater than the average of all books ratings. The shown results are truncated.

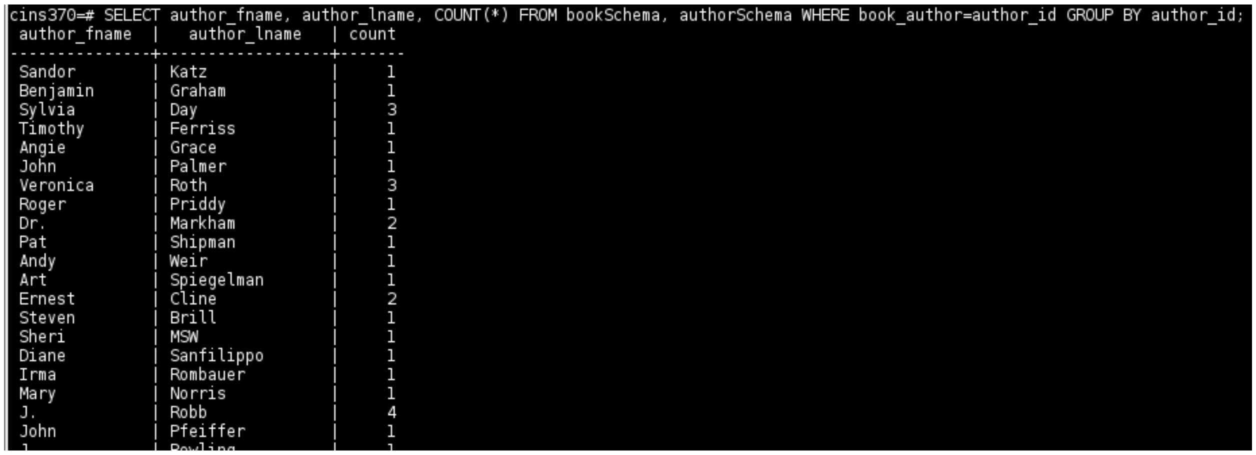


```
cins370=# SELECT book_title, avg(review_rating) FROM bookSchema, reviewSchema WHERE book_id=review_book AND review_rating >
cins370-# (SELECT avg(review_rating) from reviewSchema)
cins370-# GROUP BY book_id;
                                                                                          book_title
                                                 |       avg
--------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------+--------------------
 Great Food Fast (Best of the Best Presents) Bob Warden's Ultimate Pressure Cooker Recipes
                                                 | 4.0000000000000000
 StrengthsFinder 2.0
                                                 | 4.2857142857142857
 Our Kids: The American Dream in Crisis
                                                 | 4.1818181818181818
 White Heat 25
                                                 | 4.0000000000000000
 Business Law: Text and Cases
                                                 | 3.7857142857142857
 The 13th Disciple: A Spiritual Adventure
                                                 | 3.9230769230769231
 The Flavor Bible: The Essential Guide to Culinary Creativity, Based on the Wisdom of America's Most Imaginative Chefs
                                                 | 4.1176470588235294
 The War That Forged a Nation: Why the Civil War Still Matters
                                                 | 3.7500000000000000
 Goddesses Never Age: The Secret Prescription for Radiance, Vitality, and Well-Being
                                                 | 4.3846153846153846
 Nice Is Just a Place in France: How to Win at Basically Everything
```

```sql
SELECT author_fname, author_lname, COUNT(*)
FROM bookSchema, authorSchema
```

```sql
WHERE book_author=author_id
GROUP BY author_id;
```

This query returns the authors name and how many books they've written. The shown results have been truncated.



```
cins370=# SELECT author_fname, author_lname, COUNT(*) FROM bookSchema, authorSchema WHERE book_author=author_id GROUP BY author_id;
 author_fname  |   author_lname    | count
---------------+-------------------+-------
 Sandor        | Katz              |     1
 Benjamin      | Graham            |     1
 Sylvia        | Day               |     3
 Timothy       | Ferriss           |     1
 Angie         | Grace             |     1
 John          | Palmer            |     1
 Veronica      | Roth              |     3
 Roger         | Priddy            |     1
 Dr.           | Markham           |     2
 Pat           | Shipman           |     1
 Andy          | Weir              |     1
 Art           | Spiegelman        |     1
 Ernest        | Cline             |     2
 Steven        | Brill             |     1
 Sheri         | MSW               |     1
 Diane         | Sanfilippo        |     1
 Irma          | Rombauer          |     1
 Mary          | Norris            |     1
 J.            | Robb              |     4
 John          | Pfeiffer          |     1
```

We created a stored procedure and trigger that aid in calculating books prices. The trigger occurs before an insert or update to the stockSchema relation and only when stock_book_price is null or equal to zero. The procedure when triggered will add one to the stock_mark_up and multiply this by the stock_book_cost and store the result in stock_book_price.

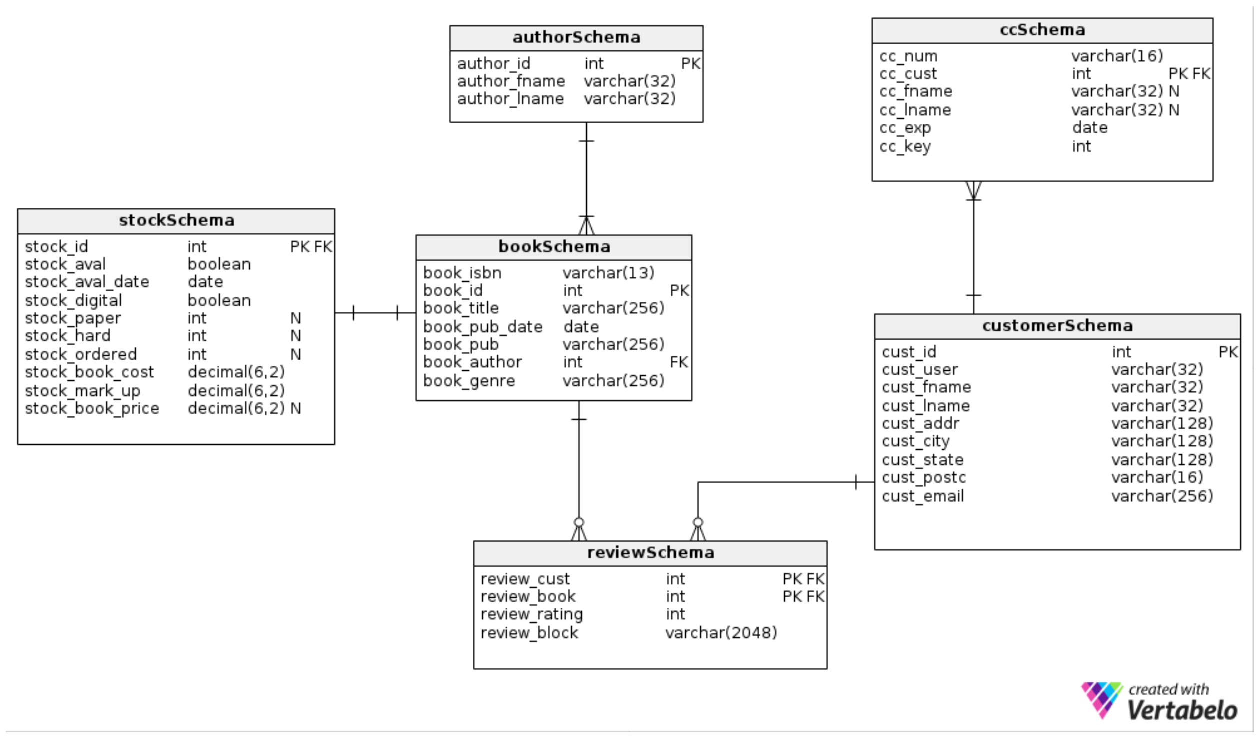
We created a single view that displays a summary based on book publishers. The view displays the number of books for each publisher as well as the average cost, mark up, price, and rating which can be sort depending on what data you want to extract from the view.

Testing:

After spending some time looking into database unit testing we decided to go with manual testing. For each constraint we wrote an appropriate amount of queries that adequately covered cases that were expected to fail and ran these after each set of changes to the schemas. The thought being that if we verify what we expect to fail everything else is working as intended.

Query results were validated by using control data and executing queries whose results were known. If the tests were successful on our control data the assumption is made that it will be successful on larger amounts of the same data.

Entity Relation Diagram:



Schemata:

```
CREATE TABLE authorSchema (
        author_id int NOT NULL,
        author_fname varchar(32) NOT NULL,
        author_lname varchar(32) NOT NULL,
        UNIQUE(author_fname, author_lname),
        PRIMARY KEY(author_id)
);

CREATE TABLE bookSchema (
        book_isbn varchar(13) NOT NULL UNIQUE,
        book_id int NOT NULL,
        book_title varchar(256) NOT NULL,
        book_pub_date date NOT NULL,
        book_pub varchar(256) NOT NULL,
        book_author int NOT NULL,
        book_genre varchar(256) NOT NULL,
        PRIMARY KEY(book_id),
        FOREIGN KEY(book_author) REFERENCES authorSchema(author_id)
);

CREATE TABLE customerSchema (
        cust_id int NOT NULL,
        cust_user varchar(32) NOT NULL UNIQUE,
        cust_fname varchar(32) NOT NULL,
        cust_lname varchar(32) NOT NULL,
        cust_addr varchar(128) NOT NULL,
        cust_city varchar(128) NOT NULL,
        cust_postc varchar(16) NOT NULL,
        cust_email varchar(256) NOT NULL UNIQUE,
        PRIMARY KEY(cust_id)
);

CREATE TABLE ccSchema (
        cc_num varchar(16) NOT NULL,
        cc_cust int NOT NULL,
        cc_fname varchar(32) NULL,
        cc_lname varchar(32) NULL,
        cc_exp date NOT NULL CHECK(cc_exp > current_date),
        cc_key int NOT NULL CHECK(cc_key >= 0000 AND cc_key <= 9999),
        PRIMARY KEY(cc_num),
        FOREIGN KEY(cc_cust) REFERENCES customerSchema(cust_id)
);
```

```sql
CREATE TABLE reviewSchema (
       review_cust int NOT NULL,
       review_book int NOT NULL,
       review_rating int NOT NULL CHECK(review_rating >= 0 AND review_rating <= 5),
       review_text varchar(2048) NOT NULL,
       PRIMARY KEY(review_cust, review_book),
       FOREIGN KEY(review_cust) REFERENCES customerSchema(cust_id),
       FOREIGN KEY(review_book) REFERENCES bookSchema(book_id)
);

CREATE TABLE stockSchema (
       stock_id int NOT NULL,
       stock_aval boolean NOT NULL,
       stock_aval_date date NOT NULL DEFAULT current_date,
       stock_digital boolean NOT NULL,
       stock_paper int NULL DEFAULT 0,
       stock_hard int NULL DEFAULT 0,
       stock_ordered int NULL DEFAULT 0,
       stock_book_cost decimal(6, 2) NOT NULL
       stock_mark_up decimal(6, 2) NOT NULL DEFAULT 0.9 CHECK(stock_mark_up >= 0.01)
       stock_book_price decimal(6, 2) NULL,
       PRIMARY KEY(stock_id),
       FOREIGN KEY(stock_id) REFERENCES bookSchema(book_id)
);

CREATE OR REPLACE FUNCTION stock_insert() RETURNS trigger AS $$
       BEGIN
              NEW.stock_book_price := NEW.stock_book_cost * (1 + NEW.stock_mark_up);
              RETURN NEW;
       END
$$ LANGUAGE plpgsql;

CREATE TRIGGER stock_insert BEFORE INSERT OR UPDATE
ON stockSchema
FOR EACH ROW
WHEN (NEW.stock_book_price IS NULL OR NEW.stock_book_price <= 0)
EXECUTE PROCEDURE stock_insert();

CREATE VIEW summary AS
SELECT book_pub, COUNT(*), avg(stock_book_cost) AS avg_cost, avg(stock_mark_up) AS
avg_mark_up, avg(stock_book_price) AS avg_price, avg(review_rating) AS avg_rating
FROM bookSchema, stockSchema, reviewSchema
WHERE book_id=stock_id AND book_id=review_book
GROUP BY book_pub;
```

- Changes
  - Fixed appropriate data types
  - Moved the foreign key from bookSchema to stockSchema
  - Added various missing constraints, mostly unique constraints
  - Removed default from stock_book_price in stockSchema
  - Added stored procedure and trigger to replace default from stock_book_price in stockSchema
  - Added summary view

Indexes:

PostgreSQL adds indexes wherever PRIMARY KEY is used. If they weren't automatically added, that's where we would add them. We don't use many other columns that aren't primary keys in our clauses so adding indexes elsewhere wouldn't help performance very much.

Python script:

```
#! /usr/bin/env python2

# Created by: Jason Boutte
# CINS 370
# Spring 2015
# Phase 4

# Required libraries
# python-amazon-simple-product-api
#  https://github.com/yoavaviram/python-amazon-simple-product-api
# faker
#  https://github.com/joke2k/faker

# Python version 2.7.9

import sys
import time
import csv
import string
import random
import datetime
import argparse
from faker import Faker
from amazon.api import AmazonAPI
```

```
# returns the first and last token of the author
def parse_author(author):
  tokens = string.split(author, ' ')
  return (tokens[0], tokens[len(tokens)-1])

# generates an entry for stockSchema
def generate_stock(index, list_price, price):
  aval = bool(random.randint(0,1))

  aval_date = None

  # make up date for availabilty
  if not aval:
    t = datetime.date.today()
    t += datetime.timedelta(random.randint(1, 30))
    aval_date = str(t)

  digi = bool(random.randint(0, 1))

  paper = None
  hard = None
  ordered = None

  if aval:
    paper = random.randint(500, 1000)
    hard = random.randint(100, 500)
  else:
    ordered = random.randint(500, 1000)

  cost = price

  # if theres no list price we make up a mark up
  # guarantee a minimum of 0.01
  if not list_price:
    mark_up = random.random()+0.01
  else:
    # try to infer a mark_up from list_price and price
    mark_up = abs(float(list_price)/float(price)-1);
    # guarantee minimum of 0.01
    if mark_up < 0.01:
      mark_up = random.random()+0.01

  # format to 2 decimal marks
  mark_up = '%.2f'%(mark_up)
  # return stock tuple
  return (index, aval, aval_date, digi, paper, hard, ordered, cost, mark_up, 0.0)

# generates entry for bookschema
def generate_book(isbn, index, title, pub_date, pub, author, genre):
  # toss entry if theres no isbn
  if isbn == '':
    return None

  # create pub_date if it doesnt exist
  if pub_date == 'None':
    pub_date = datetime.date.today()

  return (isbn, index, title, pub_date, pub, author, genre)

# generates entry for authorschema
def generate_author(index, author):
  parsed = parse_author(author)

  return (index, parsed[0], parsed[1])

# generates entry for customerschema
def generate_customer(fake, index):
  user = fake.user_name()
  fname = fake.first_name()
  lname = fake.last_name()
  addr = fake.address()
  city = fake.city()
  state = fake.state()
  postc = fake.postcode()
  email = fake.email()

  return (index, user, fname, lname, addr, city, state, postc, email)

#generates entry for creditcard
def generate_creditcard(fake, index, fname, lname):
  cc_num = fake.credit_card_number()
  cc_exp = datetime.date.today() + datetime.timedelta(3650)
  cc_key = fake.credit_card_security_code()

  return (cc_num, index, fname, lname, cc_exp, cc_key)

#generates entry for reviewschema
def generate_review(fake, index):
  review_cus = random.randint(0, 499)
  review_rating = random.randint(0, 5)
  review_text = fake.text()

  return (review_cus, index, review_rating, review_text)

def format_csv():
  fake = Faker()

  f = open('data.csv', 'r')
  reader = csv.reader(f)

  book_dict = {}
  stock_list = []
  author_dict = {}

  book_index = 0
  author_index = 0

  print("Generating CSV files")

  # iterate over each input line
  for row in reader:
    # generate new author if not exist
    if not row[2] in author_dict:
      author_dict[row[2]] = generate_author(author_index, row[2])
      author_index += 1

    author_id = author_dict[row[2]][0]

    # generate a new book
    book = generate_book(row[0], book_index, row[1], row[3], row[4], author_id, row[5])

    # skip if theres an errro generating book
    if book == None:
      continue
    else:
      # generate stock and increment book index
      stock = generate_stock(book_index, row[6], row[7])

      book_index += 1

    # check for duplicate isbn entries, occasional bug
    if not book[0] in book_dict:
      book_dict[book[0]] = book
      stock_list.append(stock)

  # write book to csv file
  f = open('books.csv', 'wb')
  writer = csv.writer(f)

  for isbn_key, book in book_dict.iteritems():
    writer.writerow(book)

  print("Wrote " + str(len(book_dict)) + " books to books.csv")

  # write authors to csv file
  f = open('authors.csv', 'wb')
  writer = csv.writer(f)

  for author, parsed in author_dict.iteritems():
    writer.writerow(parsed)

  print("Wrote " + str(len(author_dict)) + " authors to authors.csv");

  # write stock to csv file
  f = open('stock.csv', 'wb')
  writer = csv.writer(f)

  for stock in stock_list:
    writer.writerow(stock)

  print("Wrote " + str(len(stock_list)) + " items to stock stock.csv")

  # write customer and creditcard csv files
  f = open('customers.csv', 'wb')
  writer = csv.writer(f)

  f1 = open('cc.csv', 'wb')
  cc_writer = csv.writer(f1)

  cc_count = 0

  for x in range(500):
    # generate a new customer
    customer = generate_customer(fake, x)
    writer.writerow(customer)

    # generate 0 to 2 credit cards per customer
    for y in range(random.randint(0, 2)):
      cc_count += 1
      cc = generate_creditcard(fake, x, customer[2], customer[3])
      cc_writer.writerow(cc)

  print("Wrote 500 customers to customers.csv")
  print("Wrote " + str(cc_count) + " credit cards to cc.csv")

  # write customer reviews
  f = open('reviews.csv', 'wb')
  writer = csv.writer(f)

  # guard against multiple reviews by same customer
  cust_review = {}

  # each book has a chance for reviews
  for isbn_key, book in book_dict.iteritems():
    # chance to review between 0 and 30 books
    review_num = random.randint(0, 30)

    for y in range(review_num):
      cust_id = random.randint(0, 500)

      review = generate_review(fake, book[1])

      if not (review[0], review[1]) in cust_review:
        cust_review[(review[0], review[1])] = review
        writer.writerow(review)

  print("Wrote " + str(len(cust_review)) + " customer reviews to reviews.csv")

def generate_csv():
  # initialize amazon api with access key, secret key, and associate tag
  amazon = AmazonAPI('AKIAJPT5M67Z5DB6R3XA', 'P0ekhRiDVDC2xeJa4fZz1P5qHY/B2Qig71G6wZB3',
'thedeepdark-20')

  print("Querying amazon API")
  # returns available book subjects
  subjects = amazon.browse_node_lookup(BrowseNodeId=1000)

  f = open('data.csv', 'wb')
  writer = csv.writer(f)

  print("\tReturned with " + str(len(subjects[0].children)) + " subjects")

  # creates books and author lists
  for subject in subjects:
    for genre in subject.children:
      # skip calendar entries
      if genre.name.text == 'Calendars': continue

      # returns first 1000 entries in each subject
      # Amazons api limits the number of return pages to 10
      # with 10 items on each for a maximum of 100 items
      books = amazon.search_n(100, Condition='All', BrowseNode=genre.id, SearchIndex='Books',
MaxQPS=0.9)

      print("Queried " + genre.name + ", returned " + str(len(books)) + " books")

      failed = 0

      for book in books:
        b_isbn = book.isbn
        b_title = book.title
        b_pub_date = str(book.publication_date)
        b_genre = genre.name
        b_publisher = book.publisher
        b_list_price = book.list_price[0]
        b_price = book.price_and_currency[0]

        if len(book.authors) == 0:
          break

        book_item = [b_isbn, b_title, book.authors[0], b_pub_date, b_publisher, b_genre,
b_list_price, b_price]

        for x in range(len(book_item)):
         if isinstance(book_item[x], str):
           book_item[x] = unicode(book_item[x], 'utf-8')

        try:
          writer.writerow(book_item)
        except UnicodeEncodeError:
          failed += 1

      print("\tDone processing books, failed to convert unicode characters " + str(failed) + "
times")

      time.sleep(5)

parser = argparse.ArgumentParser()
```

```
parser.add_argument('-g', action='store_true')

args = parser.parse_args()

if args.g:
  generate_csv()
else:
  format_csv()
```