|  |
| --- |
| ISTM 622-601 |
| Clustering Project - Part 1 |
| Group 1-1 |

|  |
| --- |
| Mrinalini Dey  Jathin Dhulipalla  Sreelaxmi Hegde  Harmit Jasani  Jonathan Paine  Yishen Zhang |

Contents

[1. Scope 2](#_Toc21099666)

[2. Cluster Creation 2](#_Toc21099667)

[3. Dataset information & ETL Process 7](#_Toc21099668)

[3.1 Dataset - SeattlePublicLibrary 7](#_Toc21099669)

[3.2 Data extraction, cleaning and transformation was a 2-part process 7](#_Toc21099670)

[3.2.1 Part 1: Pre-processing using Python (Jupyter Notebook) 7](#_Toc21099671)

[3.2.2 Part 2: Data Processing using Java & Talend DI 8](#_Toc21099672)

[4. Data loading, tools used, cluster connection to the tool used to load data 13](#_Toc21099673)

[5. SQL Code- Creation of Database & Table 15](#_Toc21099674)

[6. Entity Relationship Diagram 17](#_Toc21099675)

[7. Python & HTML code 18](#_Toc21099676)

[7.1API code: 18](#_Toc21099677)

[7.2 HTML code: 23](#_Toc21099678)

[8. Answering the business questions 26](#_Toc21099679)

[8.1 Home page 26](#_Toc21099680)

[8.2 Functionality 1 26](#_Toc21099681)

[8.3 Functionality 2 27](#_Toc21099682)

[8.3 Functionality 3 28](#_Toc21099683)

[9. Process & Data Flow 28](#_Toc21099684)

# Scope

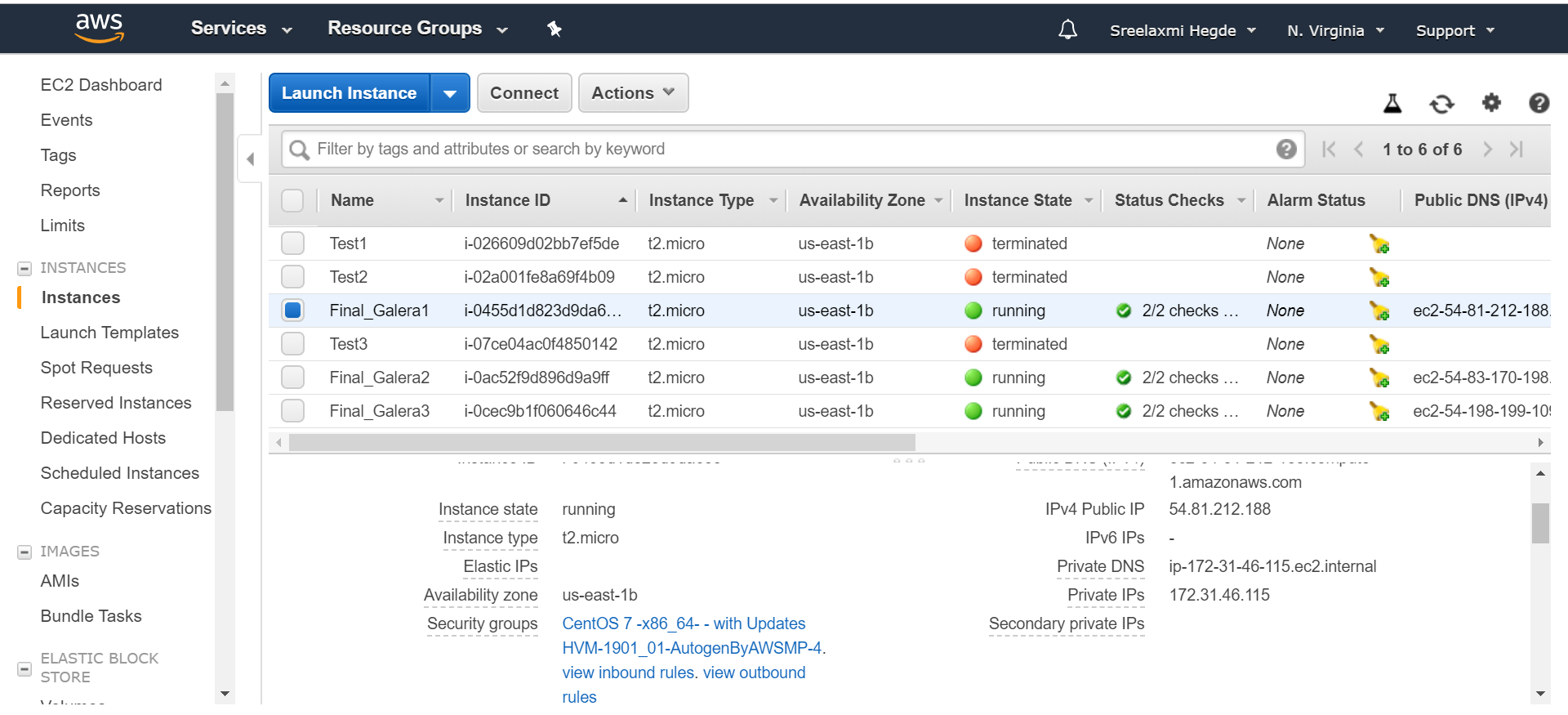
We are a consulting company helping the Seattle Public Library with building a collection management information system. In building this system we should help them with achieving their Mission and Strategic Direction. This system shall achieve these three key features

* 1. In the juvenile books category, what is the most checked out book?
     1. This will allow the library to know what is currently popular in the juvenile books category, to look into purchasing. This will help the library manage inventory efficiently.
  2. Which books in the juvenile books category haven’t been checked out in the last three years?
     1. This is to ensure we don’t stock up on JC books that aren’t popular. This would help us be more efficient with respect to using inventory space.
  3. Who is the most popular author in the juvenile books category based on the number of books checked out in the last 3 months? We should pre-order this author’s book.
     1. Based on this information, the library can organize events like fundraisers and charities collaborating with popular authors.

# Cluster Creation

* Created 3 instances using AWS EC2 console.
* We chose CentOS 7 (x86\_64) operating system distribution.
* We went with free tier eligible t2.micro instance
* In the configuration details of instances, we picked 3 instances and left all the other settings to default values
* For encryption, we generated a key pair and the .pem key was converted to .ppk (private key) using PuttyGen

Screenshots of AWS EC2 console with 3 instances named Final\_Galera 1, Final\_Galera 2, Final\_Galera 3



* After we ensured the 3 nodes were up and running, we installed and configured the database and Gallera software
* We obtained the external IP address from details of each instance on AWS and used ssh and encryption key to login to each node
* We used yum commands on each node:

sudo yum -y update

We installed firewalld as well using:

sudo yum -y install rync firewalld

These commands were run on Putty server

We used MariaDB repository to install MariaDB and chose version 10.4

* We used the lines on repository tool to copy into a new repo file, /etc/yum.repos.d/mariadb.repo.
* On each node we executed below yum command:

sudo yum install MariaDB-server MariaDB-client

Next steps are about configuring MariaDB and Galera

* In order to configure MariaDB on each node, edit /etc/my.cnf.d/server.cnf under [mysql] heading.
* Get to this file using command

Sudo vi edit /etc/my.cnf.d/server.cnf

Type “I” to insert below command under mysql heading:

datadir=/var/lib/mysql

socket=/var/lib/mysql/mysql.sock

bind-address=0.0.0.0

user=mysql

default\_storage\_engine=InnoDB

innodb\_autoinc\_lock\_mode=2

innodb\_flush\_log\_at\_trx\_commit=0

innodb\_buffer\_pool\_size=128M

binlog\_format=ROW

log-error=/var/log/mysqld.log

* Execute above command by hitting on Esc, type in :wq!
* In a similar way execute below command lines under galera heading:

Insert each of the instance’s Private IP that you see on AWS console in the below command line:

wsrep\_on=ON

wsrep\_provider=/usr/lib64/galera-4/libgalera\_smm.so

wsrep\_node\_name='galera1'

wsrep\_node\_address="**172.31.32.161**"

wsrep\_cluster\_name='galera-training'

wsrep\_cluster\_address="gcomm://**172.31.32.161**"

Security setting on nodes:

* TCP ports used by Galera are as follows:

3306 (MariaDB’s default), 4444, 4567, and 4568—and 4567 for UDP

* By executing below command line on each node, we open the ports:

sudo semanage port -a -t mysqld\_port\_t -p tcp 3306

sudo semanage port -a -t mysqld\_port\_t -p tcp 4444

sudo semanage port -a -t mysqld\_port\_t -p tcp 4567

sudo semanage port -a -t mysqld\_port\_t -p udp 4567

sudo semanage port -a -t mysqld\_port\_t -p tcp 4568

sudo semanage permissive -a mysqld\_t

* By executing below command lines, we enabled the firewall and started it:

sudo systemctl enable firewalld

sudo systemctl start firewalld

sudo firewall-cmd --zone=public --add-service=mysql --permanent

sudo firewall-cmd --zone=public --add-port=3306/tcp --permanent

sudo firewall-cmd --zone=public --add-port=4444/tcp --permanent

sudo firewall-cmd --zone=public --add-port=4567/tcp --permanent

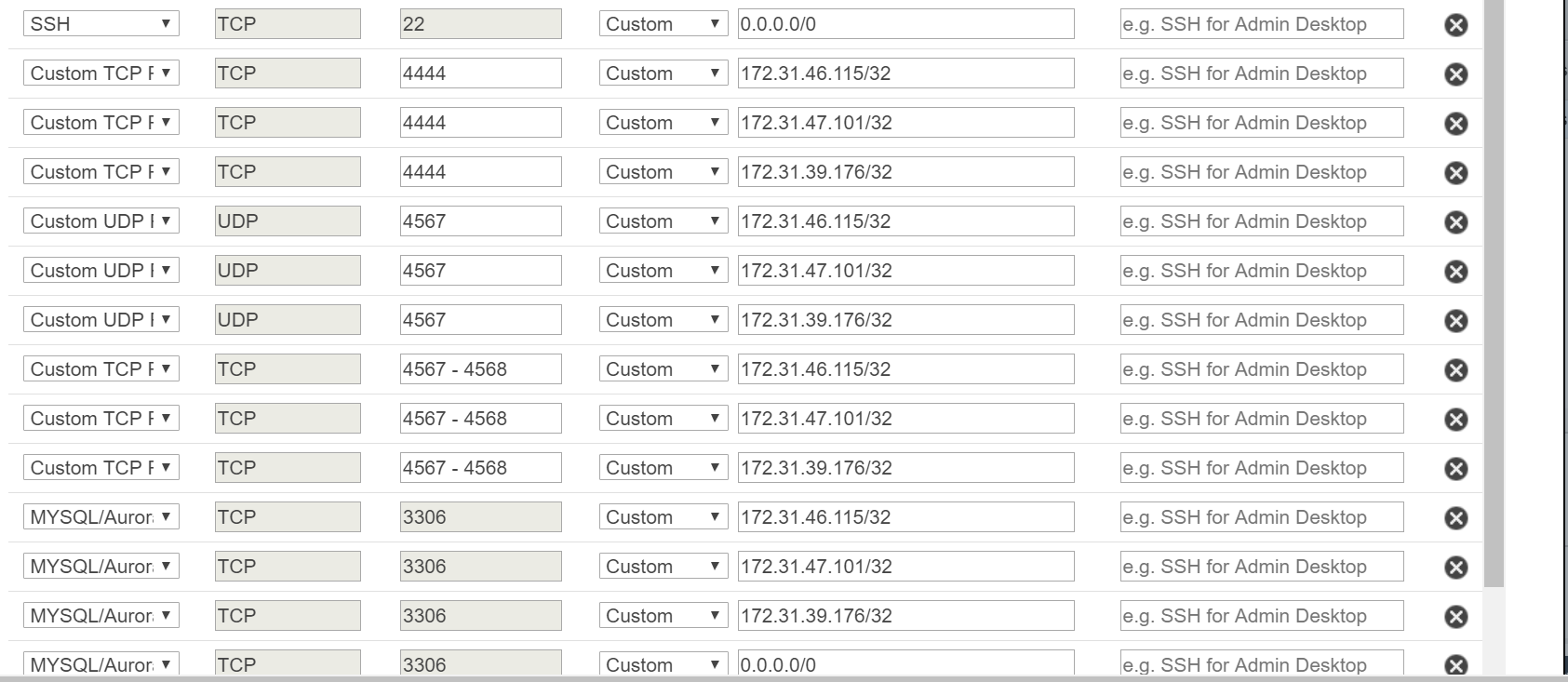
sudo firewall-cmd --zone=public --add-port=4567/udp --permanent

sudo firewall-cmd --zone=public --add-port=4568/tcp --permanent

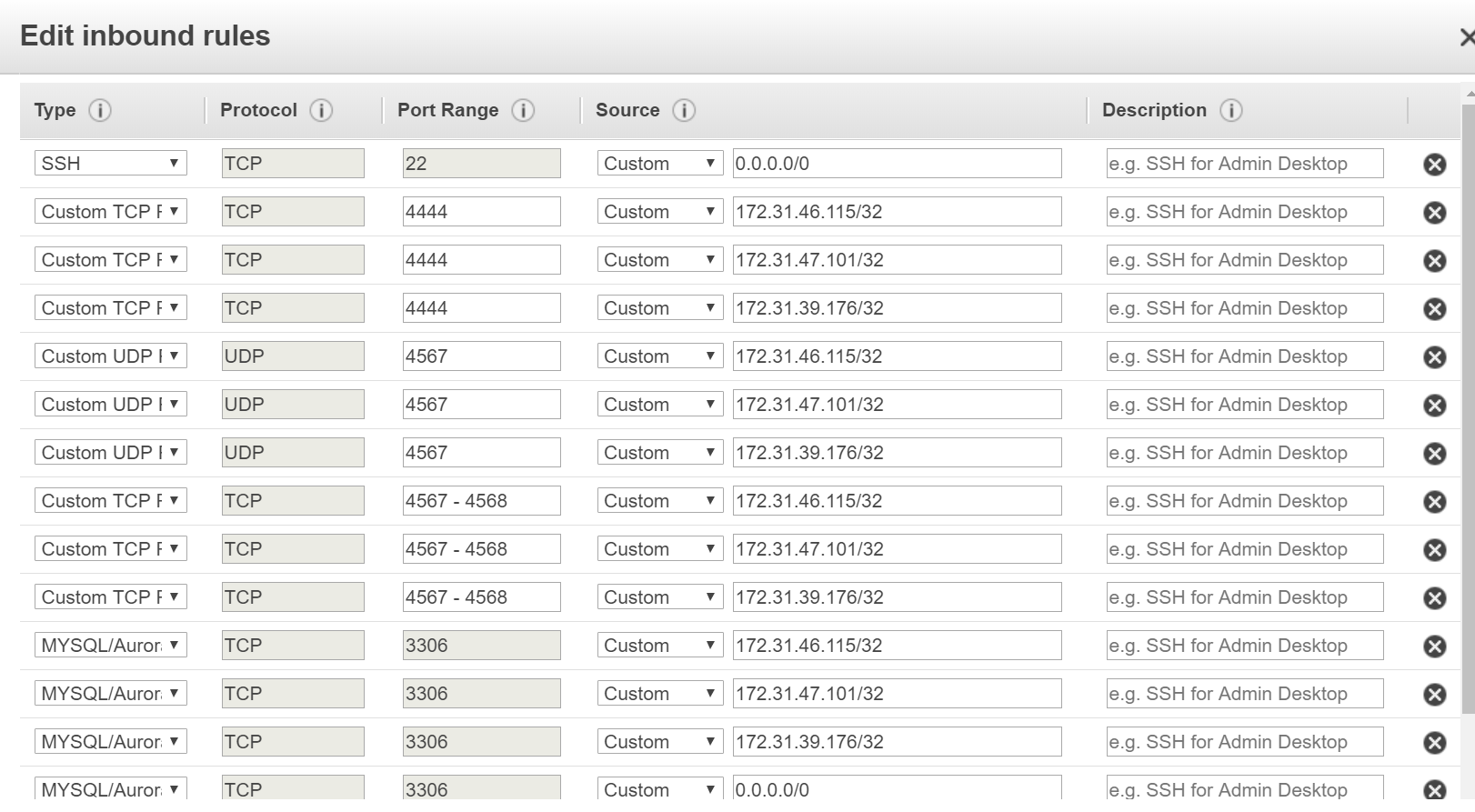
sudo firewall-cmd --reload

* In order for each node to interact with AWS instances, the below settings were configured on AWS by navigating to Security Group -> Inbound -> Edit.
* Below screenshot is of our AWS instances’ security settings

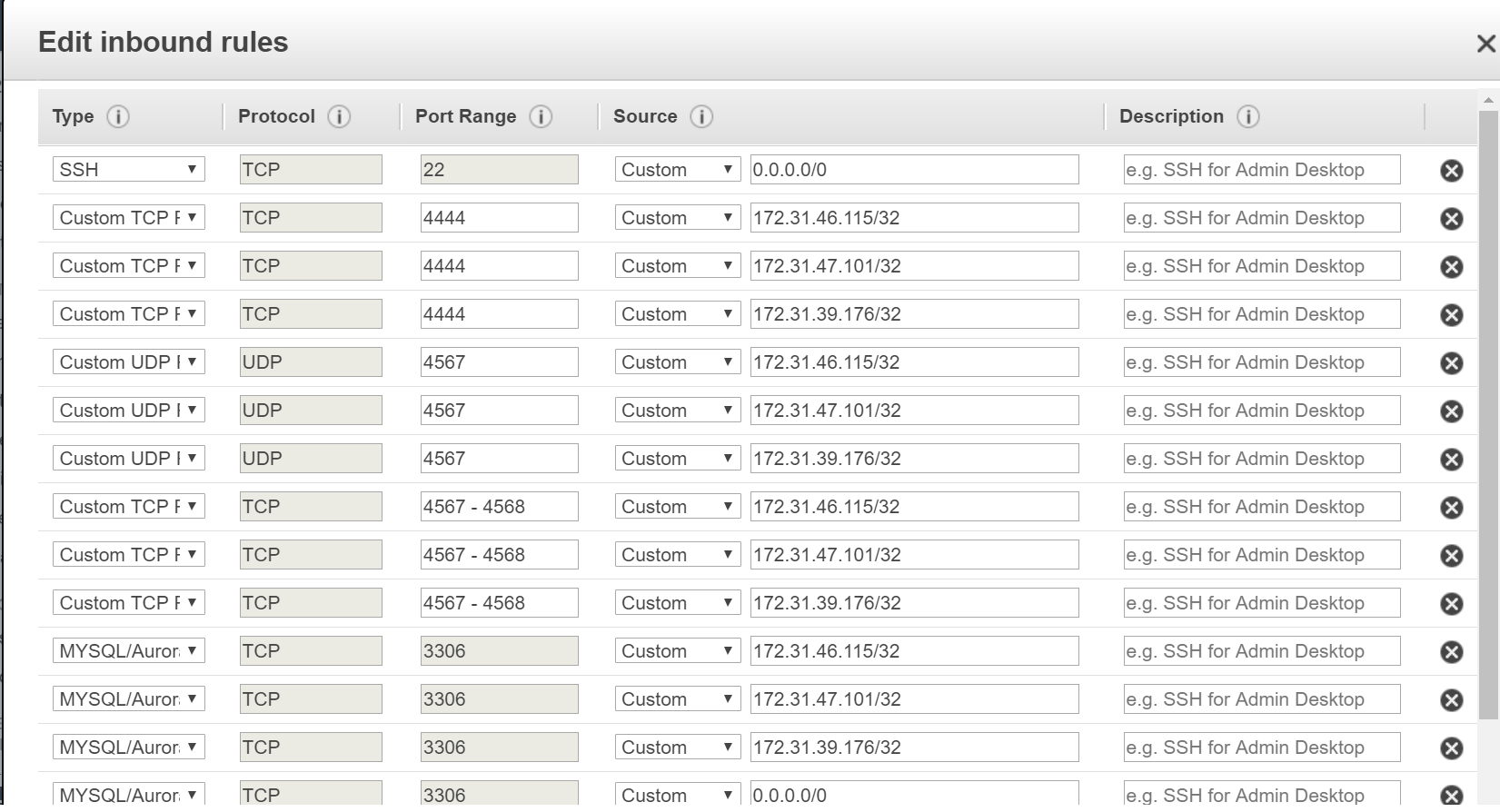
Instance 1:



Instance 2:



Instance 3:



* We started the first instance by running below script

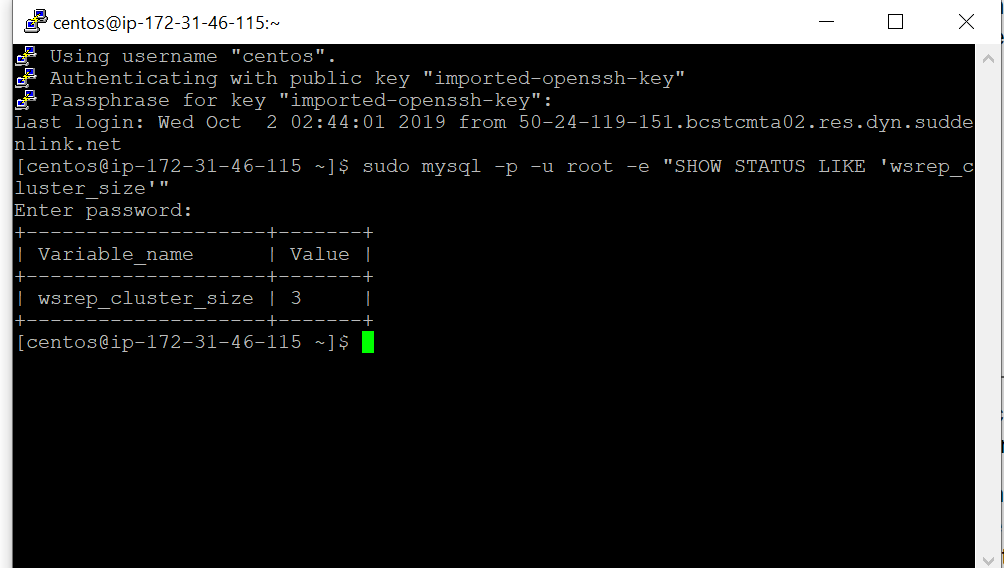
sudo galera\_new\_cluster

* On other nodes, executed below command line:

sudo systemctl start mariadb

Below command line displayed the number of nodes that were up and running:

sudo mysql -p -u root -e "SHOW STATUS LIKE 'wsrep\_cluster\_size'"

And the result/value column displayed 3 as seen in the screenshot below

# Dataset information & ETL Process

## 3.1 Dataset - SeattlePublicLibrary

Checkouts by Title (Physical)

Inventory

Data Dictionary

## 3.2 Data extraction, cleaning and transformation was a 2-part process

### 3.2.1 Part 1: Pre-processing using Python (Jupyter Notebook)

Description: Read & modify delimiters using Python for both inventory and checkouts datasets

import pandas as pd

df\_checkouts = pd.read\_csv("E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/Old Checkouts\_By\_Title\_Data\_Lens.csv")

df\_checkouts.dropna()

print(df\_checkouts.head(20))

# export checkouts data

export\_csv\_checkouts = df\_checkouts.to\_csv (r'E:\0\_TAMU - MS MIS - Mays Business School\Semester 3\Advanced Database Management\Team Project\to be cleaned\checkouts\old\_df\_checkouts\_input.csv', index = None, header=True, sep="^")

df = pd.read\_csv("E:/0\_TAMU - MS MIS - Mays Business School/Semester 3/Advanced Database Management/Team Project/to be cleaned/Library\_Collection\_Inventory.csv")

df.dropna()

print(df.head(20))

# export inventory data

export\_csv = df.to\_csv (r'E:\0\_TAMU - MS MIS - Mays Business School\Semester 3\Advanced Database Management\Team Project\to be cleaned\export\_dataframe.csv', index = None, header=True, sep="^")

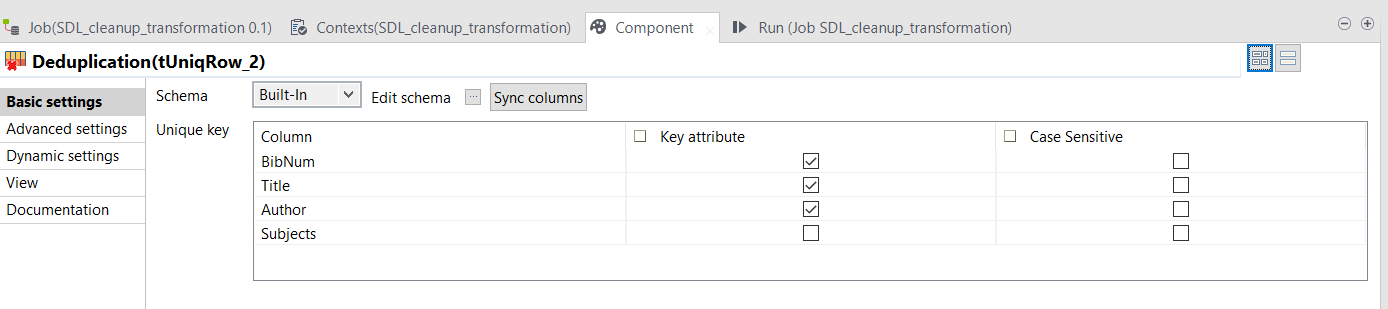
### 3.2.2 Part 2: Data Processing using Java & Talend DI

**Description:** Operations performed -

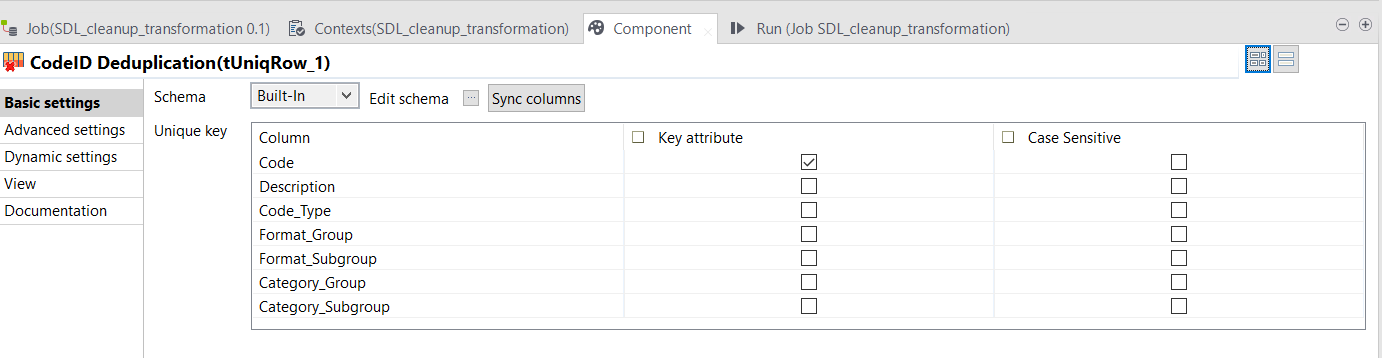
1. Extraction
2. Deduplication
3. Regex cleaning
4. Transformation
5. Normalization

Talend Components Used:

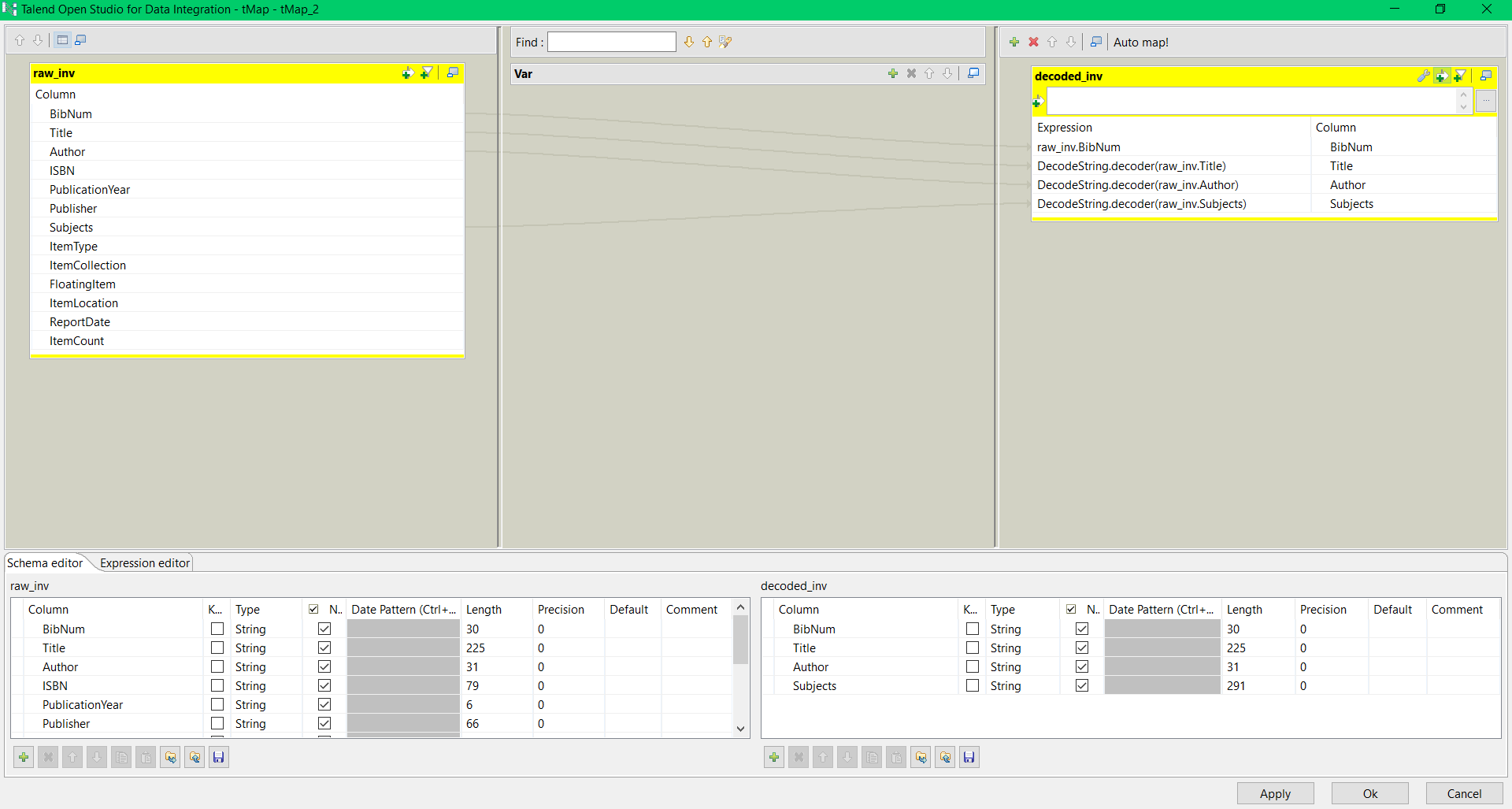
1. **tFileInputDelimited:** Configures delimited metadata for input file.
   1. SDL\_Inventory\_cleanup
   2. SDL\_Data\_Dictionary\_cleanup
   3. SDL\_Checkouts\_cleanup
   4. Normalized SDL\_Checkouts
   5. Cleaned\_SDL\_Inventory
2. **tFileOutputDelimited:** Configures delimited file for output records
   1. Cleaned\_SDL\_Inventory
   2. Cleaned\_SDL\_Data\_Dictionary
   3. Normalized SDL\_Checkouts
   4. Invalid SDL\_Checkouts
   5. Valid Checkouts
   6. Invalid Checkouts
3. **tUniqRow:** Deduplicates input rows using specified keys
   1. Deduplication



* 1. CodeID Deduplication



1. **tMap:** Maps, modifies, filters & allows most kinds of transformational operations
   1. Decode\_Normalize



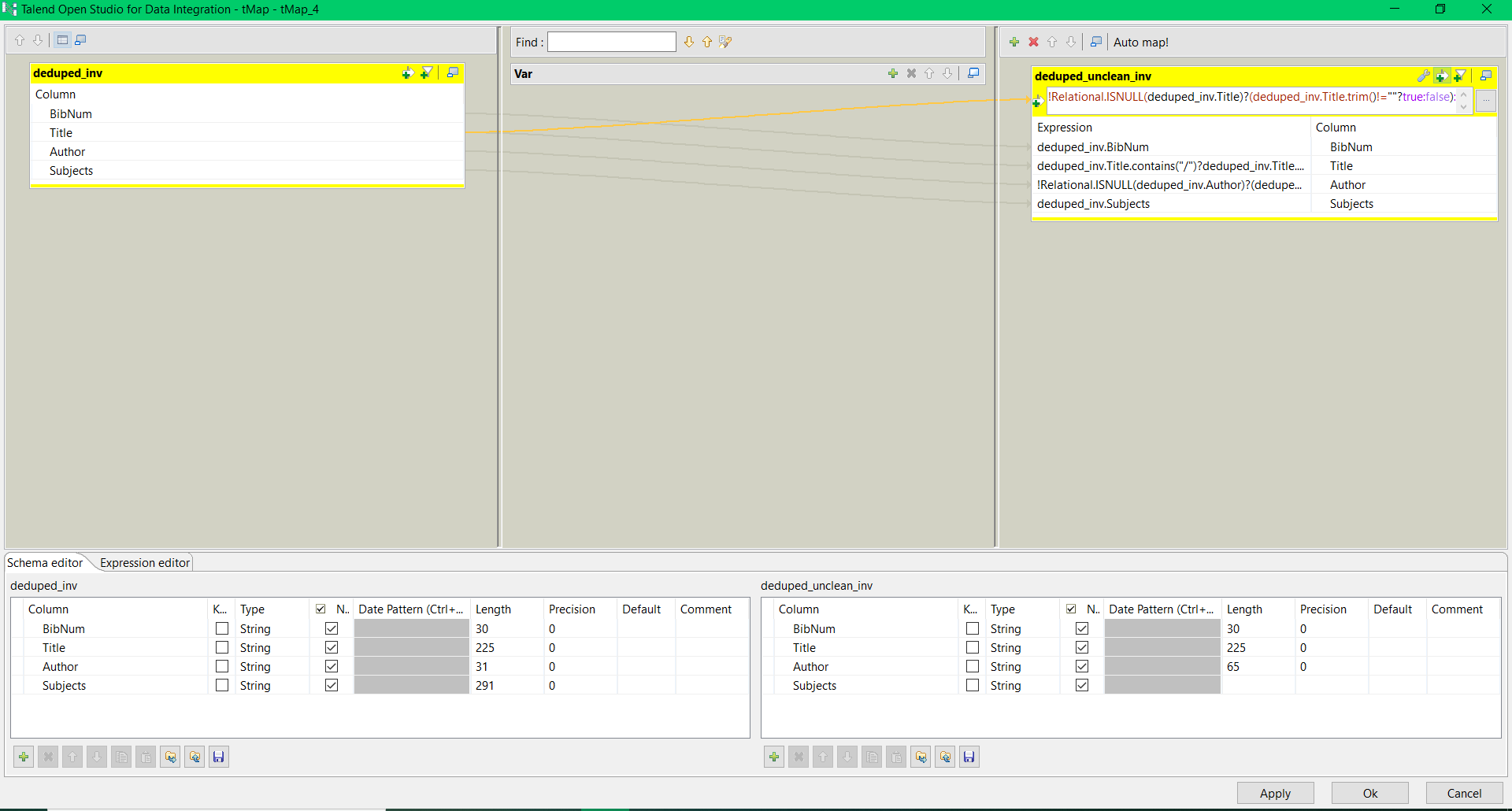
**Title decode expression logic**: DecodeString.decoder(raw\_inv.Title)

**Author decode expression logic:** DecodeString.decoder(raw\_inv.Author)

**Subjects decode expression logic:** DecodeString.decoder(raw\_inv.Subjects)

**Note:** DecodeString.decoder(input\_parameter)has been defined under routines.

* 1. Regex\_Cleanup



**Expression Filter Expression:**

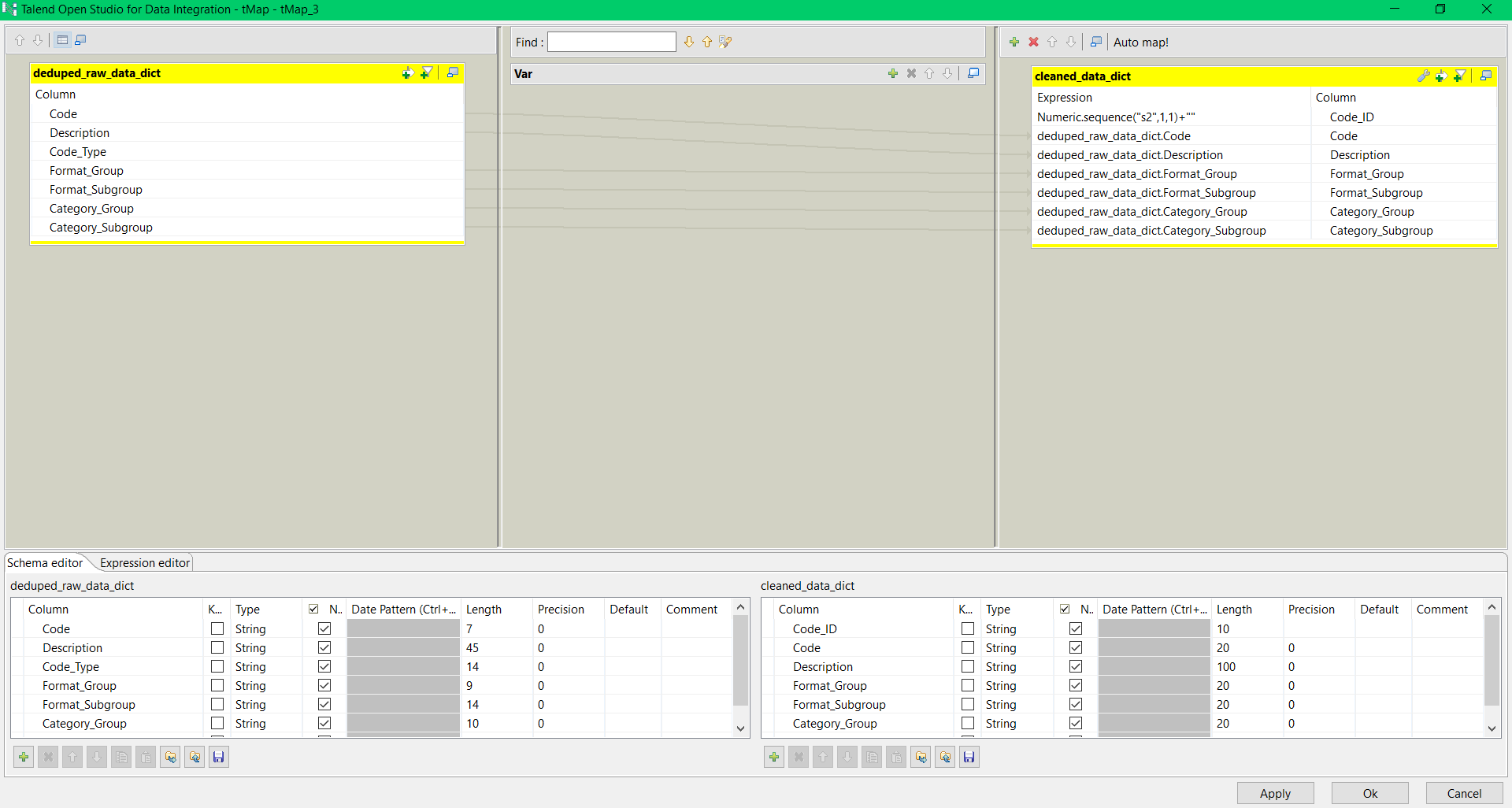
!Relational.ISNULL(deduped\_inv.Title)?(deduped\_inv.Title.trim()!=""?true:false):false

**Title Regex Cleanup Expression:** deduped\_inv.Title.contains("/")?deduped\_inv.Title.substring(0,StringHandling.INDEX(deduped\_inv.Title,"/")).trim():deduped\_inv.Title.trim()

**Author Regex Cleanup Expression:**

!Relational.ISNULL(deduped\_inv.Author)?(deduped\_inv.Author.trim()==""?"No Author Data":StringHandling.EREPLACE(deduped\_inv.Author,"([,][^A-Za-z][^A-Za-z ]+)","")):"No Author Data"

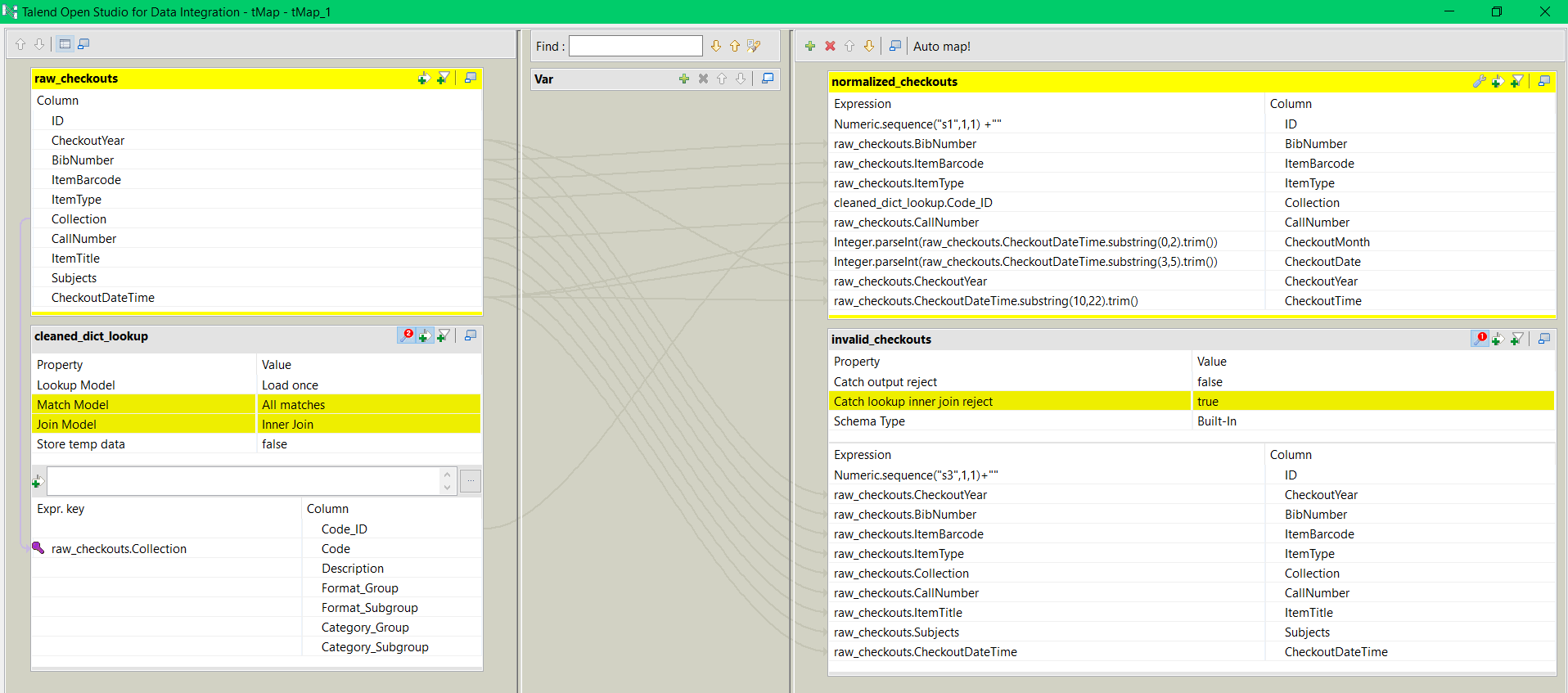
* 1. Sequence\_Generator



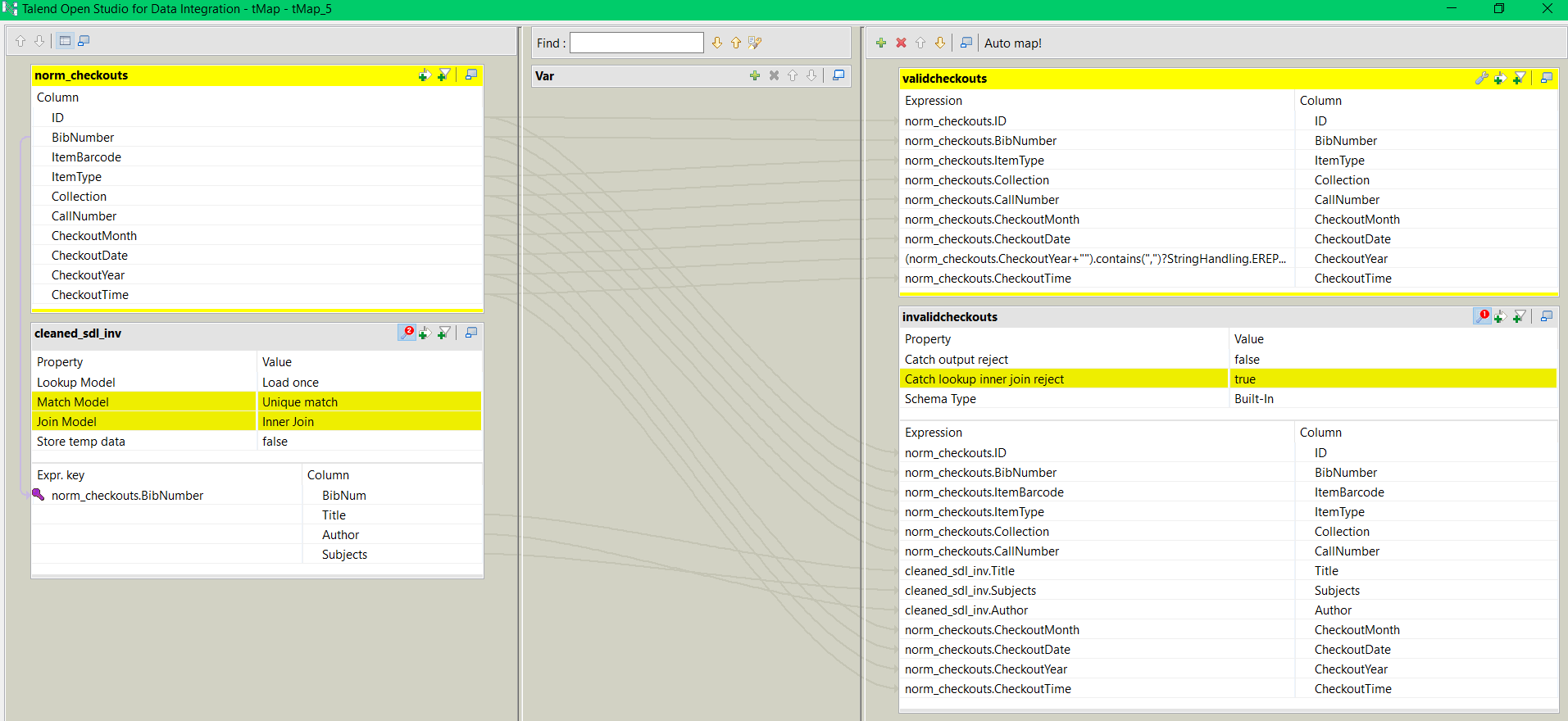
**Auto-incremented primary key for CodeID**: Numeric.sequence("s2",1,1)+""

* 1. SDL\_Checkouts Normalization

**Normalization configuration**



* 1. Inventory Join



Checkouts Year: (norm\_checkouts.CheckoutYear+"").contains(",")?StringHandling.EREPLACE((norm\_checkouts.CheckoutYear+""),",",""):(norm\_checkouts.CheckoutYear+"")

1. **tLogRow**
   1. Job\_Health\_Statistics
2. **tStatCatcher**
   1. Job\_Statistics

**Routine Used:**

**package** routines**;**

**import** java.text.Normalizer;

**public** **class** DecodeString {

**public** **static** String decoder(String input) {

String convertedString = Normalizer.*normalize*(input, Normalizer.Form.***NFD***).replaceAll("[^\\p{ASCII}]", "");

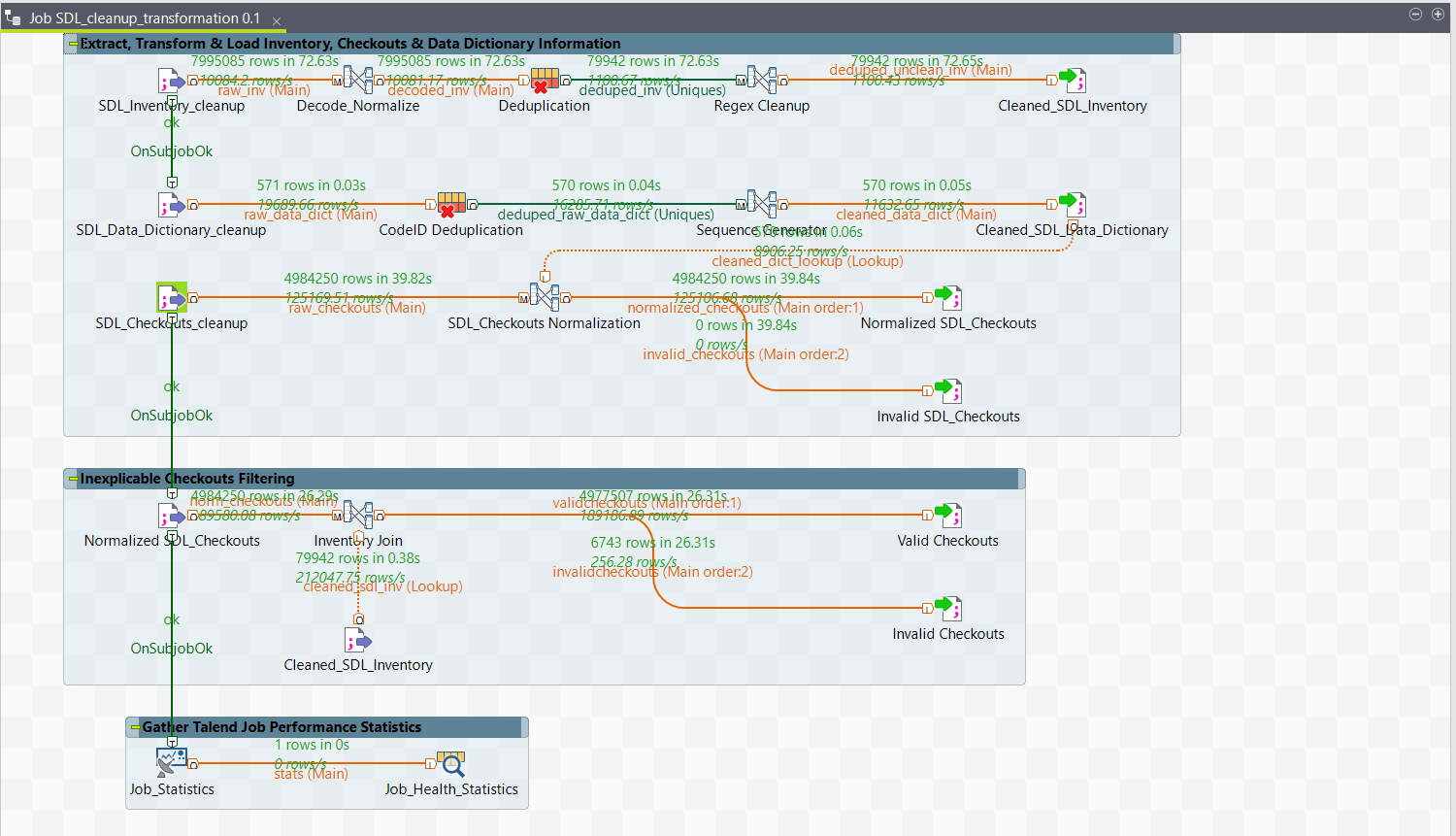
**return** convertedString;

}

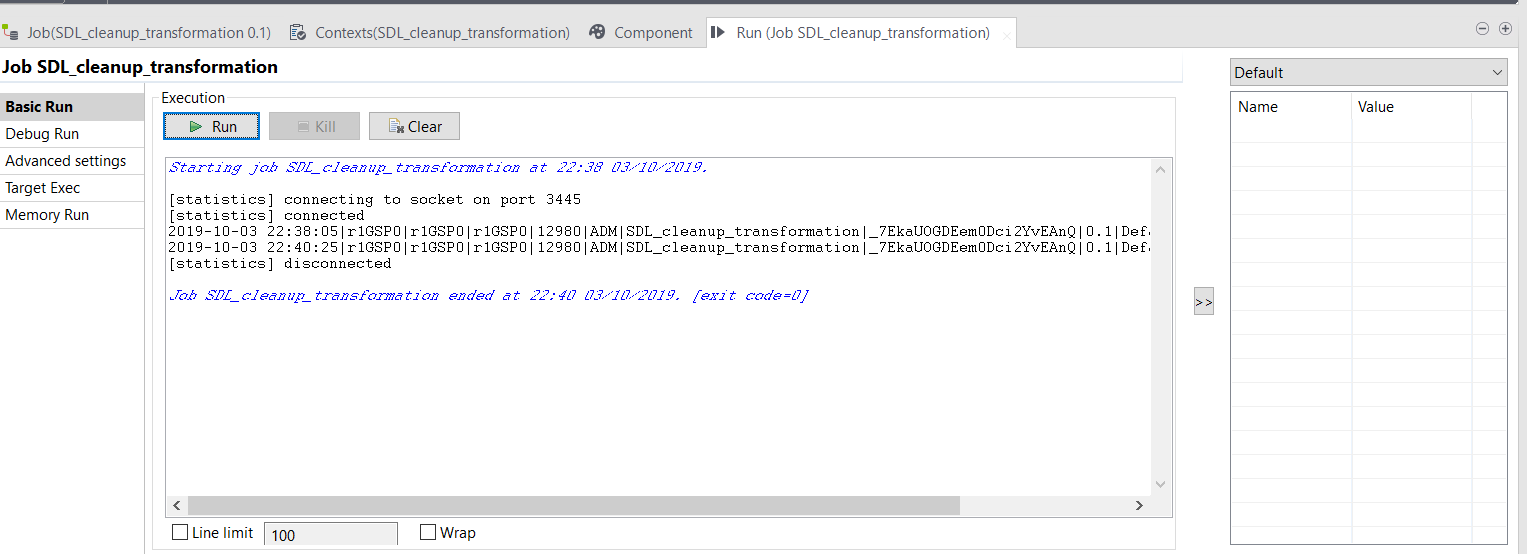
}

This function normalized special characters to make it machine-readable & UTF-8 compliant.

**Canvas Execution Area (logic flow):**



**Console:**

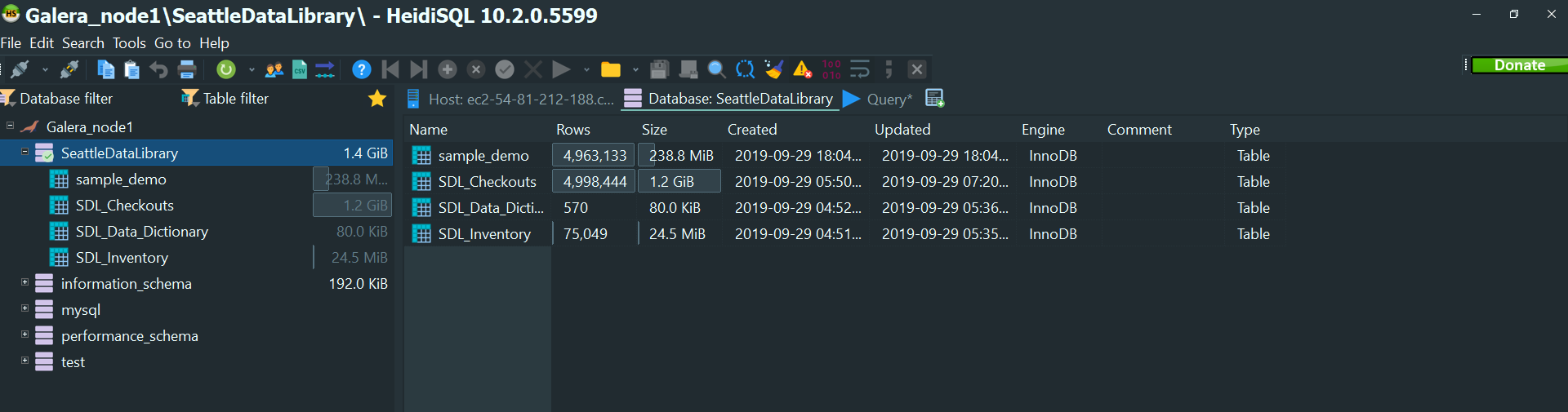


**Optimized ETL execution time:** Approximately 135 seconds

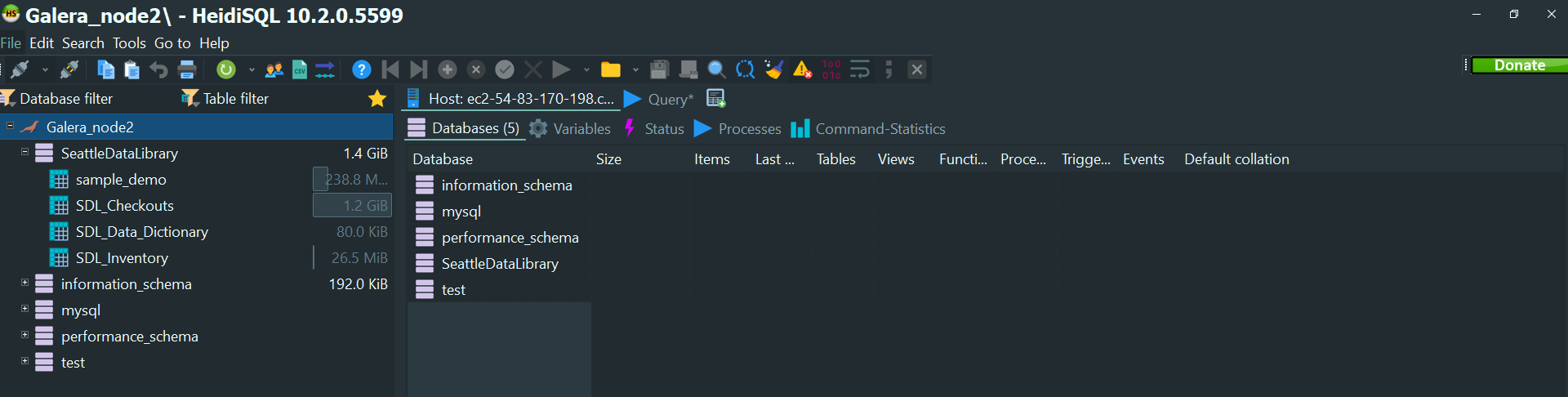
# Data loading, tools used, cluster connection to the tool used to load data

* HeidiSQL for Inventory & Data Dictionary
* Dbeaver for checkouts

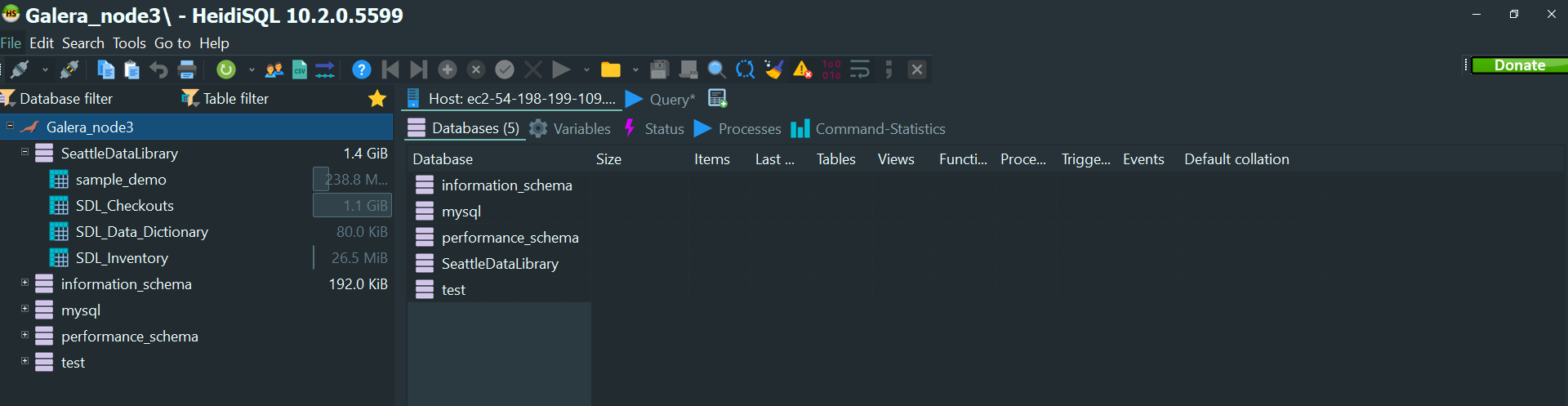
Galera node 1



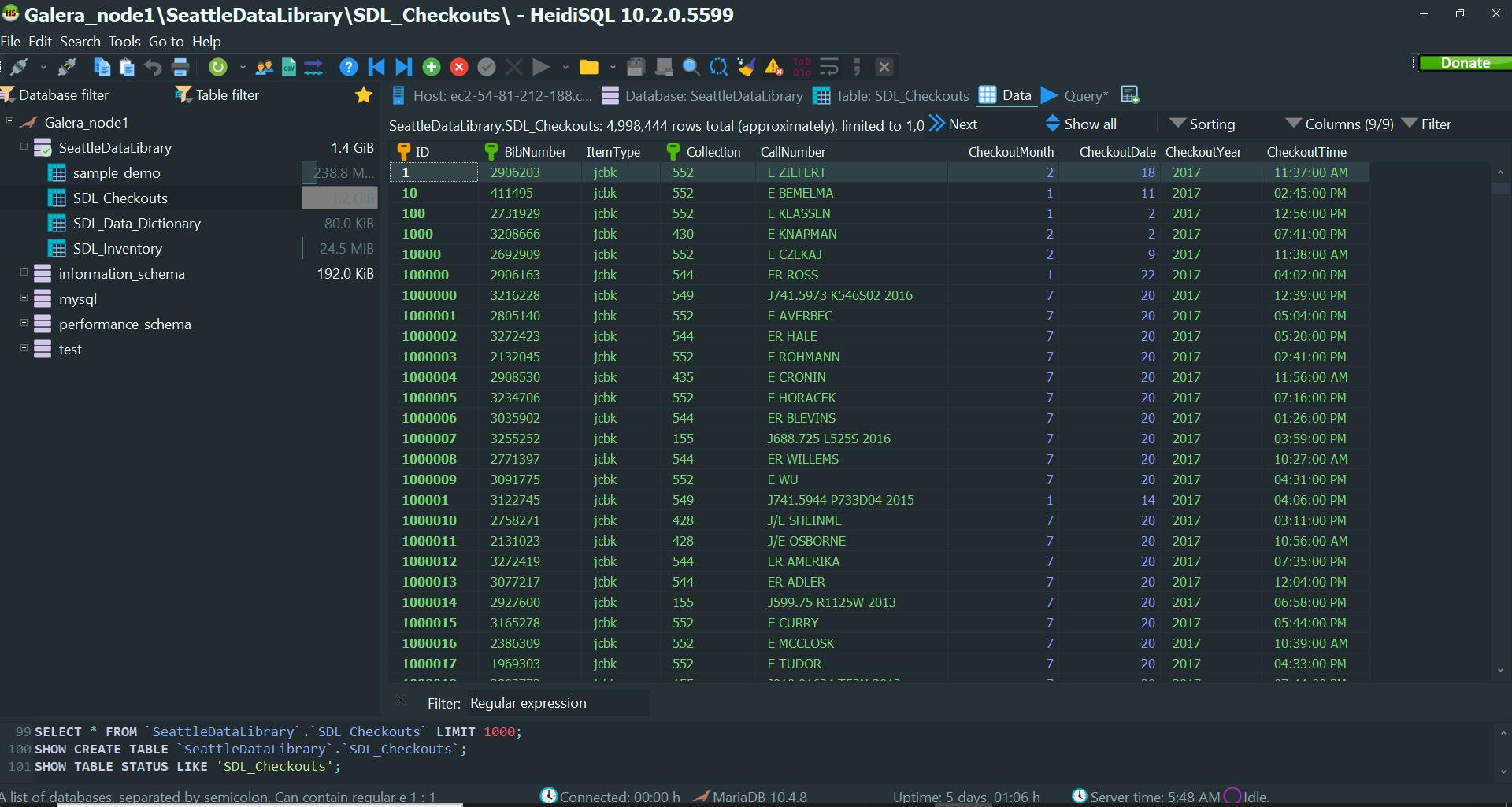
Galera node 2



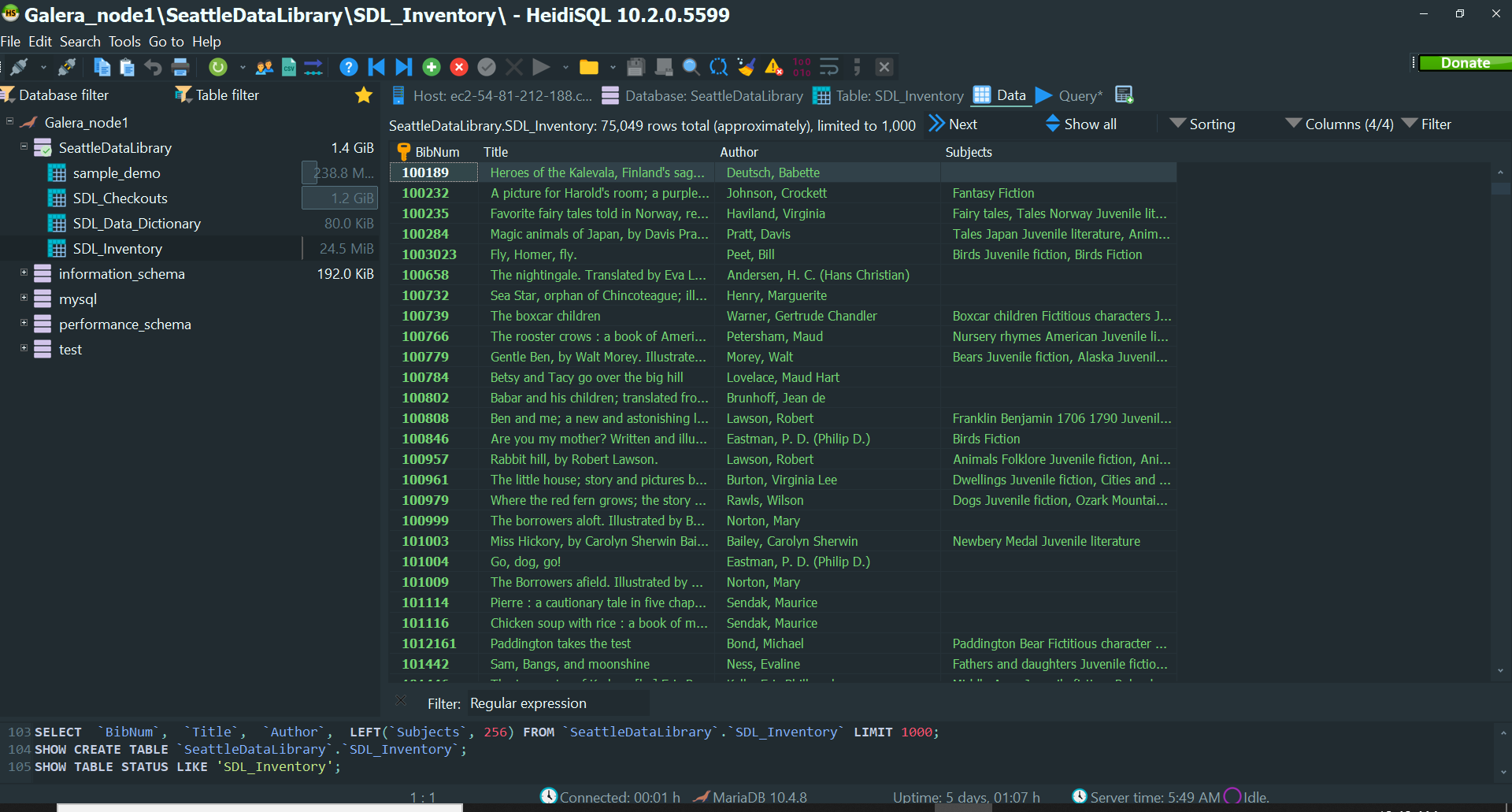
Galera node 3



Checkout Data



Inventory Data



# SQL Code- Creation of Database & Table

Creation of Database

CREATE DATABASE SeattleDataLibrary

Creation of SDL\_Checkouts table

CREATE TABLE `SDL\_Checkouts` (

`ID` VARCHAR(27) NOT NULL,

`BibNumber` VARCHAR(30) NULL DEFAULT NULL,

`ItemType` VARCHAR(10) NULL DEFAULT NULL,

`Collection` VARCHAR(20) NULL DEFAULT NULL,

`CallNumber` VARCHAR(50) NULL DEFAULT NULL,

`CheckoutMonth` INT(5) NULL DEFAULT NULL,

`CheckoutDate` INT(5) NULL DEFAULT NULL,

`CheckoutYear` VARCHAR(5) NULL DEFAULT NULL,

`CheckoutTime` VARCHAR(15) NULL DEFAULT NULL,

PRIMARY KEY (`ID`),

INDEX `FK\_SDL\_Checkouts\_SDL\_Data\_Dictionary` (`Collection`),

INDEX `FK\_SDL\_Checkouts\_SDL\_Inventory` (`BibNumber`),

CONSTRAINT `FK\_SDL\_Checkouts\_SDL\_Data\_Dictionary` FOREIGN KEY (`Collection`) REFERENCES `SDL\_Data\_Dictionary` (`Code\_ID`),

CONSTRAINT `FK\_SDL\_Checkouts\_SDL\_Inventory` FOREIGN KEY (`BibNumber`) REFERENCES `SDL\_Inventory` (`BibNum`)

)

COLLATE='latin1\_swedish\_ci'

ENGINE=InnoDB

;

Creation of SDL\_Data\_Dictionary

CREATE TABLE `SDL\_Data\_Dictionary` (

`Code\_ID` VARCHAR(10) NOT NULL,

`Code` VARCHAR(20) NULL DEFAULT NULL,

`Description` VARCHAR(100) NULL DEFAULT NULL,

`Format\_Group` VARCHAR(20) NULL DEFAULT NULL,

`Format\_Subgroup` VARCHAR(20) NULL DEFAULT NULL,

`Category\_Group` VARCHAR(20) NULL DEFAULT NULL,

`Category\_Subgroup` VARCHAR(20) NULL DEFAULT NULL,

PRIMARY KEY (`Code\_ID`)

)

COLLATE='latin1\_swedish\_ci'

ENGINE=InnoDB

;

Creation of SDL\_Inventory

CREATE TABLE `SDL\_Inventory` (

`BibNum` VARCHAR(30) NOT NULL,

`Title` VARCHAR(225) NULL DEFAULT '',

`Author` VARCHAR(65) NULL DEFAULT '',

`Subjects` VARCHAR(500) NULL DEFAULT '',

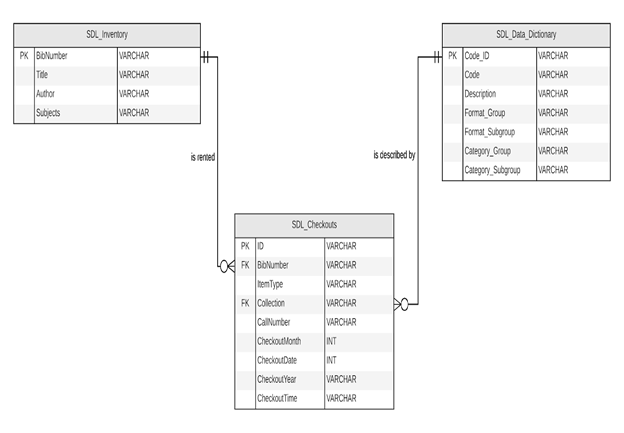
PRIMARY KEY (`BibNum`)

)

COLLATE='latin1\_swedish\_ci'

ENGINE=InnoDB;

# Entity Relationship Diagram

******

We decided to scale down the volume of our data by removing some of the columns that were not necessary. The Data Dictionary table provides information about the collection of each book. The Primary Key for the SDL\_Checkouts is an auto-incremented key that we generated in Talend. The original primary key, Record ID, was a concatenation of the date and timestamp, which we removed and replaced with this new ID for facilitating lightweight joins. The Primary Key is the BibNumber in the SDL\_Inventory table, since we got rid of the duplicate records by eliminating columns such as location and floating.

There are 2 Foreign Keys in our database, as denoted above in the ERD.

# Python & HTML code

## 7.1API code:

# importing all the required libraries and functions

import flask

from flask import Flask, render\_template, request, jsonify, redirect, url\_for

import json

import mysql.connector as mariadb

# Assigning the application to variable app

app = flask.Flask(\_\_name\_\_)

app.config["DEBUG"] = True

# This is the default route. We use get and post methods

@app.route("/", methods=["GET", "POST"])

# function defined in the default route is queries

def queries():

# The line below gets fetches and displays the html page titled index.html

render\_template("index.html")

# The html page has text boxes to get inputs from user

if request.method == 'POST':

# Extracting the required inputs from the form

booksnumber = request.form.get("booksnumber")

authorsnumber = request.form.get("authorsnumber")

# These if else statement check which api call to make based on user inputs

if booksnumber != "":

return redirect(url\_for('api\_mostcheckedout', topn=booksnumber))

elif authorsnumber != "":

return redirect(url\_for('api\_topauthors', atopn=authorsnumber))

else:

return redirect(url\_for('api\_nocheckouts'))

return render\_template("index.html")

# api for the first query. topn is a variable passed to this.

@app.route('/mostchekedout/<topn>', methods=['GET'])

def api\_mostcheckedout(topn):

# query\_parameters = request.args

# number = query\_parameters.get(topn)

# This is the sql query that we are going to pass to the database

query = ("SELECT a.BibNumber, b.Title, COUNT(a.BibNumber) "

"FROM SDL\_Checkouts a, SDL\_Inventory b "

"WHERE a.BibNumber = b.BibNum "

"GROUP BY a.BibNumber ORDER BY COUNT(a.BibNumber) DESC "

"LIMIT ")

query = query + str(topn) + ";"

# The lines below take care of connecting to the required database

connection = mariadb.connect(host='54.81.212.188',

database='SeattleDataLibrary',

user='harmitjasani@tamu.edu',

password='ADB\_ProjectRoxX')

cursor = connection.cursor()

# This line executes the query and the lines below convert them into json

cursor.execute(query)

results = json.dumps(cursor.fetchall())

results\_json = json.loads(results)

# The following block of code displays the query results in the form of an html page

results\_html = """

<!DOCTYPE html>

<html lang="en">

<head>

<title>Group 1: ISTM 622-601</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>

</head>

<body>

<div class="container">

<table class="table table-striped"><br>

<h2>Business Question 1 - Most checked out books</h2>

<tr><td><b>BibNumber</b></td><td><b>Title</b></td><td><b>Count of Checkouts</b></td></tr>

"""

for i in results\_json:

results\_html = results\_html + "<tr>" \

"<td>" + str(i[0]) + "</td>" \

"<td>" + str(i[1]) + "</td>" \

"<td>" + str(i[2]) + "</td>" \

"</tr>"

results\_html = results\_html + "</div>" \

"</table>" \

"</body>" \

"</html>"

return results\_html

# The explanation is similar to as in the above query. The only changes are for the sql query

@app.route('/nochekouts', methods=['GET'])

def api\_nocheckouts():

query = "SELECT TITLE FROM SDL\_Inventory " \

"WHERE BibNum NOT IN (SELECT DISTINCT a.BibNumber FROM SDL\_Checkouts a) " \

"AND TRIM(TITLE)<>'';"

connection = mariadb.connect(host='54.81.212.188',

database='SeattleDataLibrary',

user='harmitjasani@tamu.edu',

password='ADB\_ProjectRoxX')

cursor = connection.cursor()

cursor.execute(query)

results = json.dumps(cursor.fetchall())

results\_json = json.loads(results)

results\_html = """

<!DOCTYPE html>

<html lang="en">

<head>

<title>Group 1: ISTM 622-601</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>

</head>

<body>

<div class="container">

<table class="table table-striped"><br>

<h2>Business Question 2 - Books that have not been checked out</h2>

<tr><td><b>Title</b></td></tr>

"""

for i in results\_json:

results\_html = results\_html + "<tr>" \

"<td>" + str(i[0]) + "</td>" \

"</tr>"

results\_html = results\_html + "</div>" \

"</table>" \

"</body>" \

"</html>"

return results\_html

# The explanation is similar to as in the first query. The only changes are for the sql query

@app.route('/trendingauthors/<atopn>', methods=['GET'])

def api\_topauthors(atopn):

query\_parameters = request.args

number = query\_parameters.get('Top n Authors')

query = "SELECT b.Author, COUNT(b.Author)" \

" FROM sample\_demo a, SDL\_Inventory b" \

" WHERE a.CheckoutMonth BETWEEN (MONTH(CURDATE())- 3) AND (MONTH(CURDATE())+1)" \

" AND a.CheckoutYear = CONVERT(YEAR(CURDATE()), CHAR)" \

" AND a.BibNumber = b.BibNum" \

" AND TRIM(b.Author)<>''" \

" GROUP BY b.Author" \

" ORDER BY COUNT(b.Author) DESC" \

" LIMIT "

query = query + str(atopn) + ";"

connection = mariadb.connect(host='54.81.212.188',

database='SeattleDataLibrary',

user='harmitjasani@tamu.edu',

password='ADB\_ProjectRoxX')

cursor = connection.cursor()

cursor.execute(query)

results = json.dumps(cursor.fetchall())

results\_json = json.loads(results)

results\_html = """

<!DOCTYPE html>

<html lang="en">

<head>

<title>Group 1: ISTM 622-601</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>

</head>

<body>

<div class="container">

<table class="table table-striped"><br>

<h2>Business Question 3 - Popular Authors</h2>

<tr><td><b>Author</b></td><td><b>Count of checked out books</b></td></tr>

"""

for i in results\_json:

results\_html = results\_html + "<tr>" \

"<td>" + str(i[0]) + "</td>" \

"<td>" + str(i[1]) + "</td>" \

"</tr>"

results\_html = results\_html + "</div>" \

"</table>" \

"</body>" \

"</html>"

return results\_html

app.run()

## 7.2 HTML code:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Group 1-1: ISTM 622-601</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>

</head>

<body>

<div class="container">

<div class="jumbotron">

<center>

<h1>Seattle Library - ADM</h1>

<p>WebApp: Group 1-1 (Advanced Data Management ISTM 622-601)</p>

</center>

</div>

<form action="/" method="POST">

<div class="col-md-4 card bg-info text-white">

<center>

<h2 class="card-title">Most checked out books</h2>

<div class="card-body">

<p>These are the most checked out books. <br>How many books do you wish to see?</p>

</div>

<br>

<p>Number of required books: <br><input type = 'text' name = "booksnumber"</p>

<br><br>

<input type="submit" class="btn btn-success" value="Submit">

</center>

</div>

<div class="col-md-4 card bg-success text-white">

<center>

<h2 class="card-title">No Recent Checkout</h2>

<div class="card-body">

<p>Books that have not been checked out in the past 3 years.</p>

</div>

<br>

<input type="submit" class="btn btn-success" value="Submit">

</center>

<br>

</div>

<div class="col-md-1"></div>

<div class="col-md-4 card bg-warning text-white">

<center>

<h2 class="card-title">Trending Authors</h2>

<div class="card-body">

<p>This is a list of authors whose books were checked out most in the previous 3 months.<br>

Input required number of required records below.

</p>

</div>

<br>

<p>Number of required records: <input type = 'text' name = "authorsnumber"</p>

<br><br>

<input type="submit" class="btn btn-success" value="Submit">

</center>

<br>

</div>

</form>

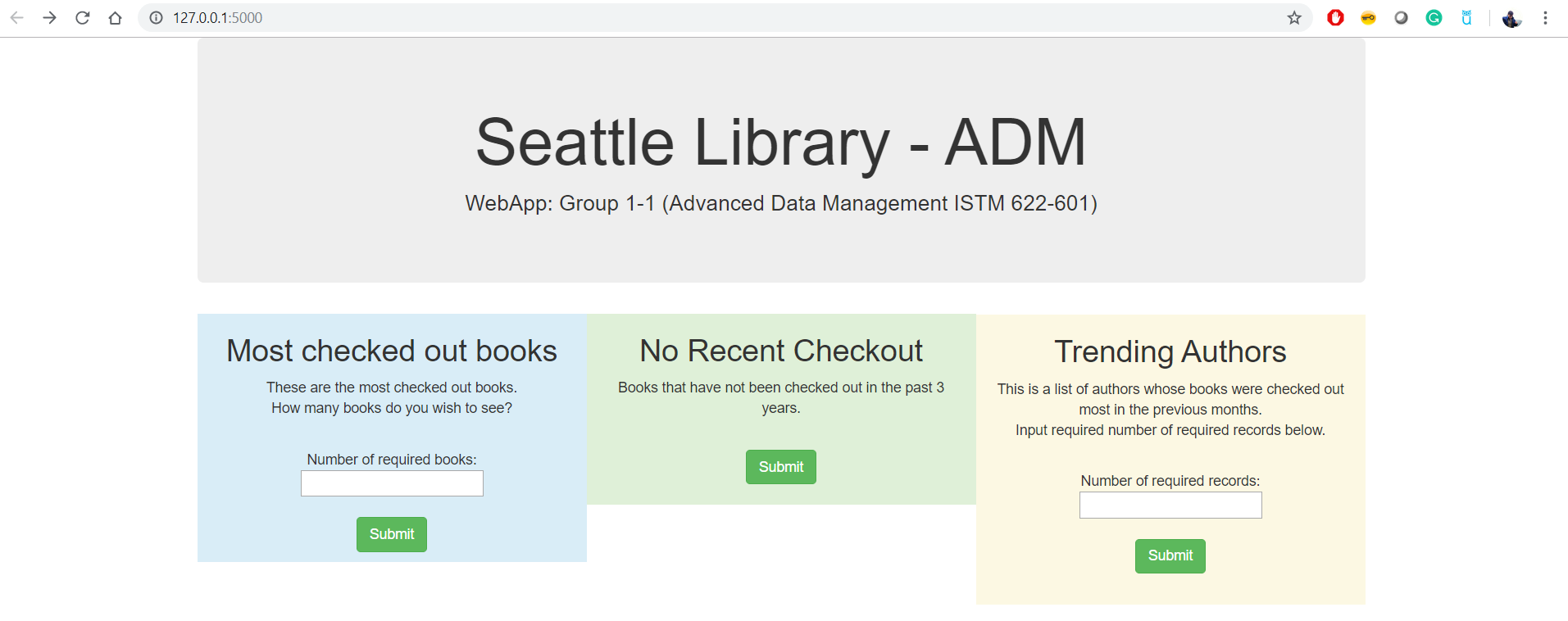
</div>

</body>

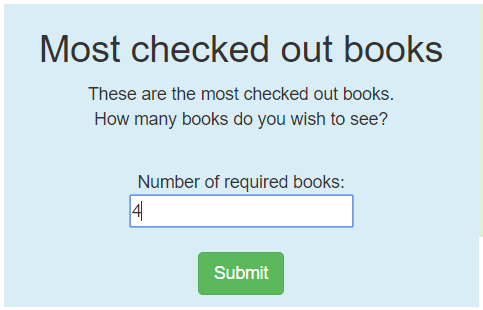
</html>

# Answering the business questions

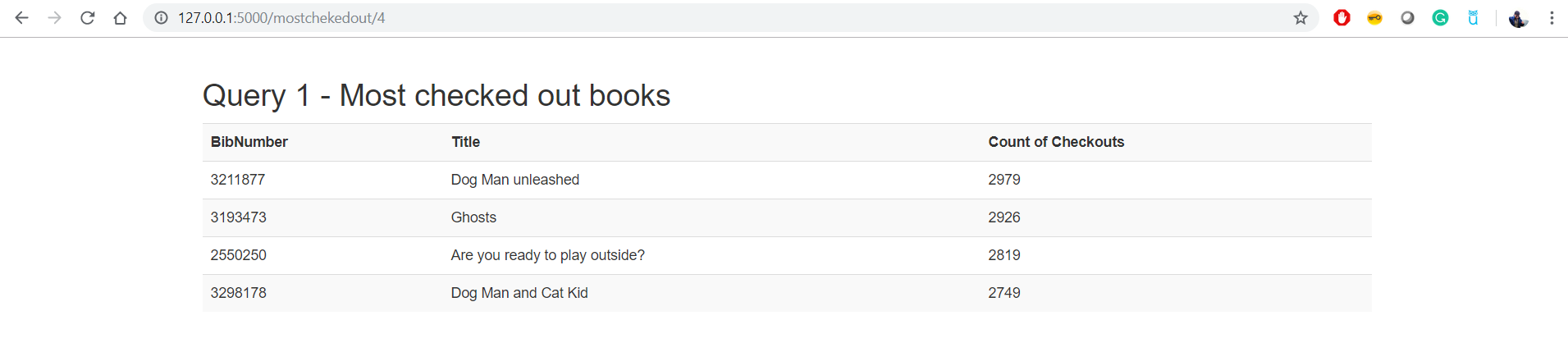
## 8.1 Home page



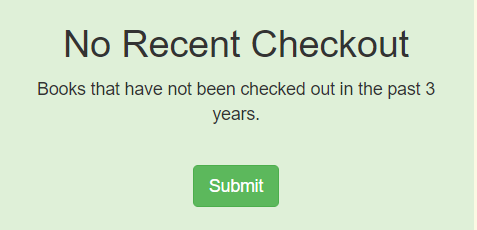
## 8.2 Functionality 1



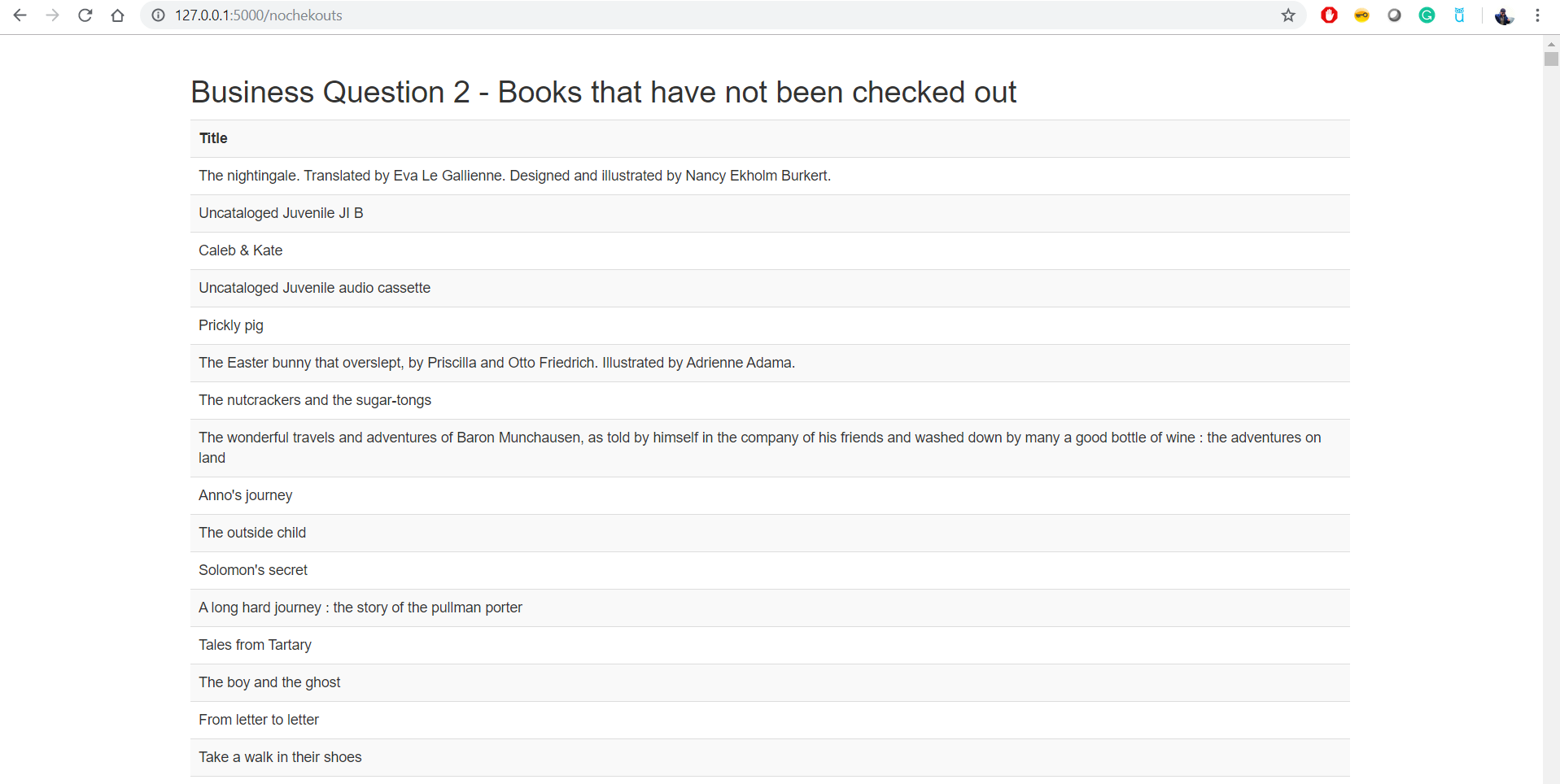
Queried results -



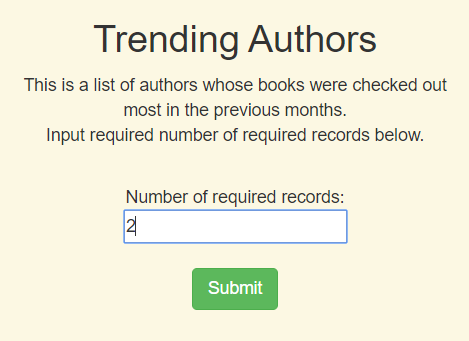
## 8.3 Functionality 2



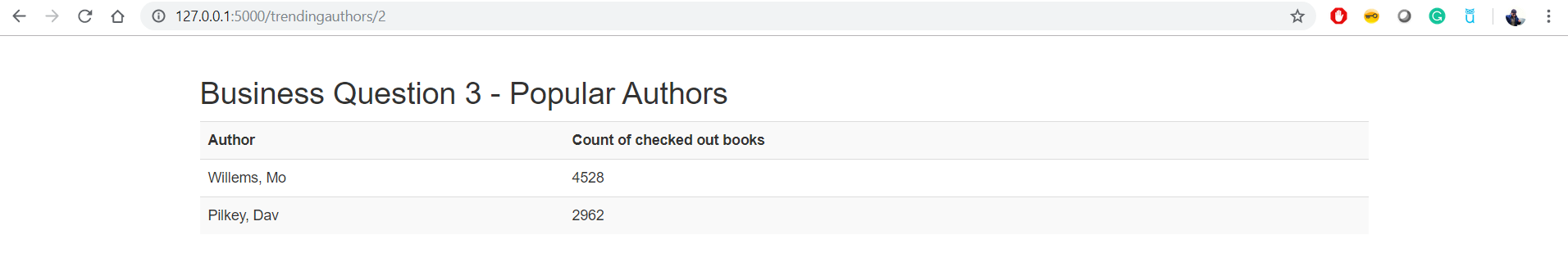
Queried results –



## 8.3 Functionality 3



Queried results –



# Process & Data Flow

****