



Master of Science

Computing and Data Analytics

MCDA 5540: Managing and Programming Databases

Project Report

Submitted by

Shubham Chumber(A00433064)

Nishant Malhotra(A00430215)

Sachit Jain(A00432721)

Submitted to

Trishla Shah

Table of contents

1. Executive summary.....	3
2. Objective	3
3. About Data.....	4
4. Methodology.....	6
4.1 Database Visualization.....	6
-EER Diagram	
-Relational Schema	
-Normalization	
4.2 Data processing and movement.....	8
- Transferring JSON to Collections	
- Data Cleaning using Python	
- Transferring data from Mongo to MySQL	
5. Conclusion.....	19
6. Future Work.....	19
7. Appendix A - [References]	19
8. Appendix B - [SQL scripts].....	20
9. Appendix C - [Web Application Screenshot].....	24
10. Appendix D - [data2mongo.sh script].....	30
11. Appendix E - [cleanJSON.py python script].....	32
12. Appendix F - [mongo2sql.sh script].....	37

1. Executive Summary

In this project we aim to tackle the problem of rectifying a Library Management system that stores information regarding the Library resources and its monthly expenditure. Our goal is to come up with an effective Database design in backend to query, store and access resources and information. A good database schema is essential as it helps in maintaining data integrity, consistency and removes the possibility of redundant data and ambiguity. In order to complete the undertaken tasks, we assumed the role of a Database Administrator first in order to come up with an effective architecture, then as a Data engineer to clean data and alter unstructured data and then finally a application developer to come up with the Web based Management System.

2. Objective

Our intent in this Project is to work on two varying Database schemas : RDBMS and NoSQL and stream data from collections into relational tables. In order to accomplish this task, we focused on conceptualizing a detailed database design using Entity Relationship diagram. We established entities, their relationship, attributes of entities and constraints. We mapped these entities into their corresponding tables, ensuring all constraints, depicting relationships via defining foreign keys and creating any additional table for relations (such as for M:N cardinality). Normalization was carried out for the above tables to further improve the table structures. The first Normal form allowed to ensure that all columns have atomic values. The second and third Normal forms allowed us to remove any partial and transitive dependencies. This helped us achieve the goal of removing any data duplication, ensure consistency of data across tables and hence remove redundancy.

As part of the project, we had to create collections which are used to store data in NoSQL databases such as MongoDB in the form of documents and fields. The data to be stored was in the JSON(JavaScript Object Notation) documents. Each JSON object has property and text fields, with different objects having altering pairs of text properties. These JSON after being processed are loaded into the Mongo database using Bash Scripts. These scripts run directly on the command line and allows us to automate tasks. We made use of Bash scripting to run SQL scripts in MySQL RDBMS, creating collections in MongoDB and importing data into respective collections. Another Bash was scripted which automated the data transfer between Mongo database to MySQL, however in order for transference of data, another script was written in Python. This script parsed the data stored and staged data to be passed. The JSON objects were later stored in a .TSV file and then the data was fed into MySQL to insert data into tables. Finally, a web application was developed in order simulate the Library management system which allows the user to view all the tables in database, register for the management system, add an article pertaining to a magazine view, add and cancel transactions. This

application was made .NET MVC framework. The backend database server used is MYSQL. Styling elements and enhancing UI was carried out via CSS and Bootstrap.

3. About Data

3.1 Database

The Current information system consists of all the data pertaining to Authors and their books and Journals. However, we need to optimize this structure as it is not optimized for Journal. This is because each Scientific journal has a large multitude of articles ranging to many hundreds. Further, each Scientific Journal can have multiples editions published in a year. So we need to come up with an effective way to keep track of all articles information within each journal disseminated over years.

Along with extensive Journal information, we are also required to maintain constraints on the data that is kept in the tables such as any magazine must have a name assigned i.e. a tuple cannot have Null values in its field. Further we need to ensure that articles published within the same volume have exact same Publication year and that every magazine must maintain a Unique Volume field such that no two volumes can have same volume number.

We are also supposed to maintain information of the loyal Customer of the Halifax Science Library. Since the terms of loyal customer are ambiguous, we have made the assumption that it is those customers which have been visiting the Library for the past five years, as we are supposed to maintain records of Customers over a period of five years. We need keep record of various attributes pertaining to a customer.

As a part of the customer information we need to maintain, in addition to personal information, we need to keep track of the customer's transactions. Each customer transaction needs to store the items purchased along with the price of each item. Every customer can have multiple transactions hence we need to reference the Customer Id as a foreign key to Transactions entity. The total price computed for the transaction is a function of discount code field mentioned in the customer table. The discount code in itself is a function of the total business (amount spent).

In addition to keeping track of the resources, HSL also wants keep a check of its monthly expenditures. The rent overhead is fixed for all the months in a year. Further, the employee cost also needs to be calculated which is the aggregation of the Salaries (hourly rate*hours worked).

As part of our solution, we came up with an entity - relationship Diagram in order to diagrammatically depict the various entities within the Database, the data they will hold(fields) and their connections and relationships among each other.

The ER has 11(eleven) entities namely customer, transaction, book, author, magazine, volume, articles, library items, monthly expense, rent and employee.

The customer table stores all the information pertaining to a single customer such as the customer id, name, email address and many more customer specifics. This entity is related to a Transactions entity with stores billing specifics such as the items bought, price and we have a customer id field to keep track if a particular transaction belongs to which customer. This Transaction entity is also linked to the Library item entity. This ways, transaction can refer to Library items and get detailed product information. The library entity is the super class and has a specialization of 'Book' and 'Magazine'. The specialized classes have their own fields. In addition, both the entities are linked with another entity 'Author'. In this entity we store author specific information. However, we will be using author id as the foreign key in Book and Magazine entities to relate the Library item products with the entire data for Author, as and when needed.

We have another set of identities, however they are not linked directly to the above set of entities. This section of ER is focused on calculating, estimating and querying the expenses. We have a 'monthly expense' entity which has fields for storing the electric, heat and water bills for a particular month of a year. Employees are assets to company's effective business and all of them are entitled to a monthly remuneration. So we came up with another entity 'Employee' that stores the employees SIN, name and other personal information and hourly rate. When we relate this entity to expense table, on the relation we defined two additional attributes that allow us to store the numbers of hours an employee worked and the which date. We have another entity 'rent' which stores the year and the associated monthly rent(rent for each month in a year is same).

3.2 JSON

Data for Articles table is available in JSON file. This standard format for transmitting data between applications has property: text fields. The properties are fields which allow categorization of data with text being the actual data that a property holds. The JSON file has data from multiple sources because of which it has erroneous fields, In addition to data. The JSON data has generic article related information such as authors, article heading, journal and its volume and total pages. However data also has redundant data in the ftail property which can be removed, further any article can have multiple authors, names of these authors are given in full rather than segregated on the basis of first name and last name, as required by the database. Further the data has inconsistencies such as many of the essential properties for documents are missing. We have processed and cleaned this data and handled exceptions.

3.3 About SQL Scripts

The script file has table structures for Author, Magazine and Item. In addition to create statements, the file had Insert commands to fill these respective tables. The Author table has four fields namely ID, last name and first name of author, and email address with data for 70 plus rows is provided. The Magazine table has two fields, magazine ID and name, with the constraint that name cannot be empty field. The data for this table inserts three rows. The Item table has price and id field with eight insert statements the table with as many rows.

4. Methodology

4.1 Database Visualization

4.1.1 Entity Relationship Diagram

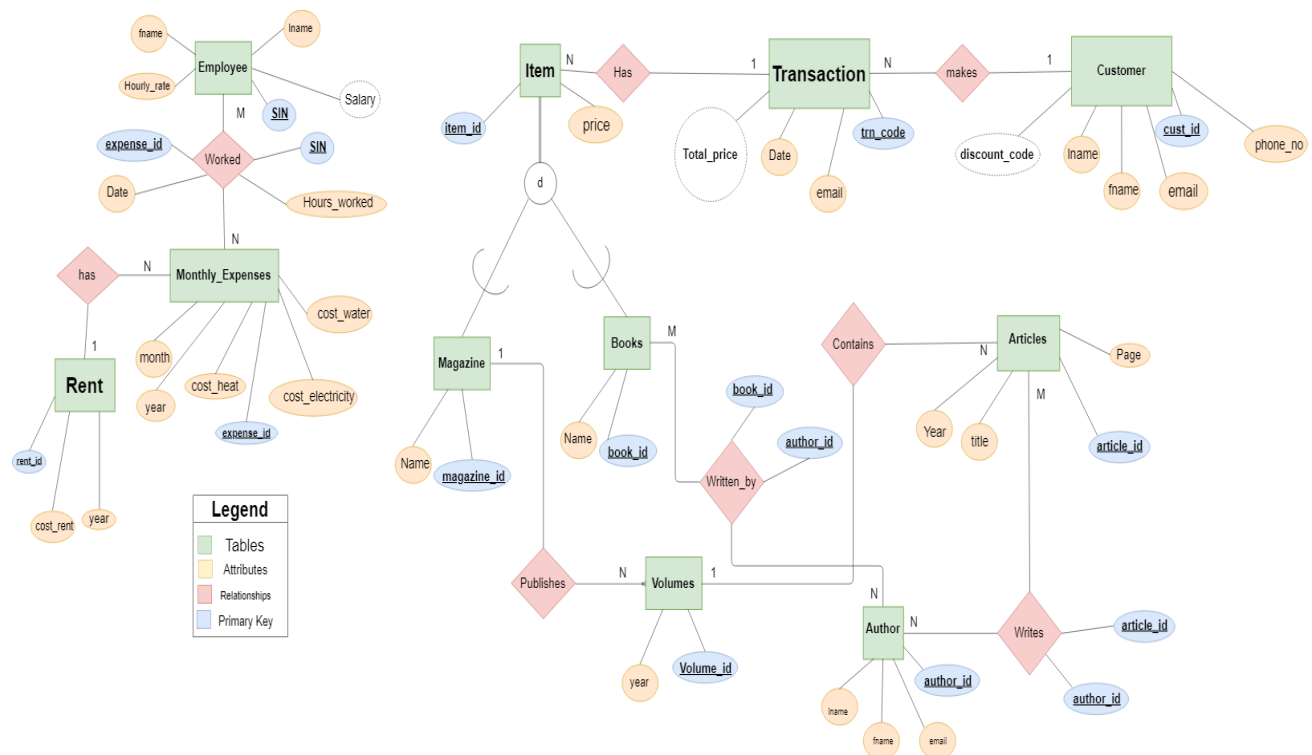
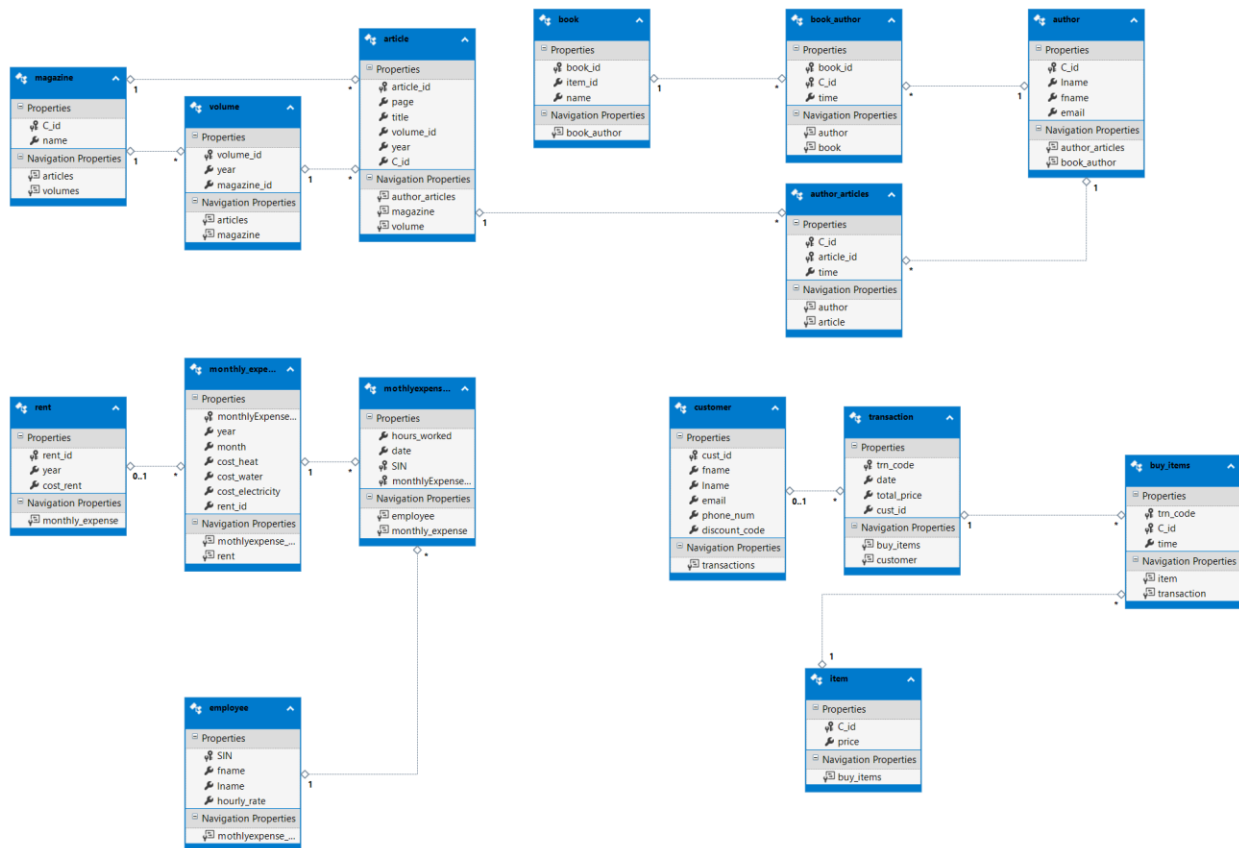


Figure 4.1.1.1 – Entities, their relationships and fields.

4.1.2 Mapping ER to Relational tables



Assumptions

- Relation between transaction and items is 1 to N, because multiple items can be found in one transaction however, multiple transaction cannot be mapped to same items as every transaction has a unique ID.
- Relation between customer and transactions in 1:N because multiple transactions can be mapped to single user however a single transaction cannot belong to multiple customers.
- Relation between Magazine and Volumes is 1:N because one magazine has multiple volumes in a single year however a unique volume cannot belong to multiple Magazines.
- Relation between is volumes and articles 1:N because one Volume can have a multitude articles however one article cannot be in multiple volumes.
- Books and Author & Author and Articles are M:N relationships, as multiple authors can contribute to multiple items(book or article) and one author can write multiple article or books.

- Relation between Monthly expense and Employee is M:N because one monthly expense consists of multiple employee (salaries) and one employee will contribute to multiple monthly expense reports.
- Relation between rent and monthly expense is 1:N as property expenditure of multiple months will depend on one Rent entry for that year.

4.1.3 Normalization

Our relational schema is Normalized to **3NF** highest normal form.

Monthly Expenses - We broke this entity into three tables -

- 1) Employee - We have added information pertaining to an individual employee. We broke employee table from monthly expense because employee can have varying working hours but fixed hour rate, further they can be part-time, full-time or contractual, so the hours worked per month vary. Hence, accommodating this info in monthly expense table would add redundancy.
- 2) Monthly Expense- This table has fixed expenses which do not change(Assumption)
- 3) Rent - We separated data into Rent table as for each month in a particular year, the rent is fixed. This allowed us to achieve consistency.

Magazine - the table was normalized as follows -

- 1) Magazine- Simple stores the Name of Magazine and its corresponding ID.
- 2) Volumes- This table stores all information for every volume of a particular Magazine. This allowed us to reduce redundancy in Magazine table.
- 3) Articles- This allowed us to reduce redundancy in volume table as all articles in a particular magazine can be referenced by Voloume_Id foreign key thus help us achieve redundancy without leading to any loss of Data.

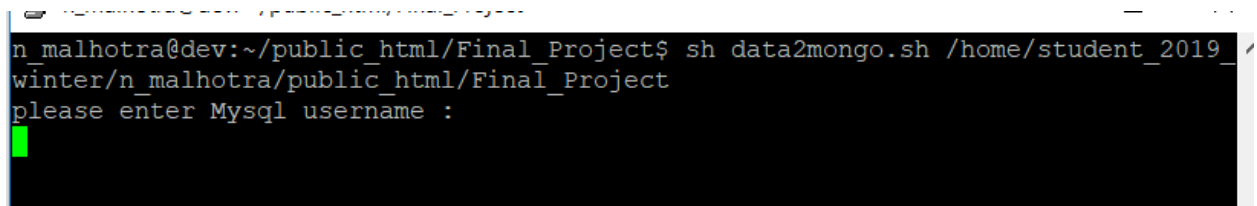
4.2 Data Processing and Movement

4.2.1. Approach -

The JSON collection is semi-structured data which was pulled from multiple sources and added to articles.json file and hence Data file is inconsistent. Issues such as missing properties, properties with string and nested JSON documents and Lists along with additional data fields which are not required in any Mongo collection. We used python script for cleaning the code. In order to come up with cleaned JSON collection and stage for MongoDB insertion, we created separate JSON files via 'cleanJSON.py' python script, for each respective Mongo collection. We parse the JSON files one document at a time, perform validations and populate 4 different output JSON files. These scripts is implicitly called by 'data2Mongo.sh' Bash script file.

4.2.2. Automating commands ‘ data2mongo.sh ’ Bash script -

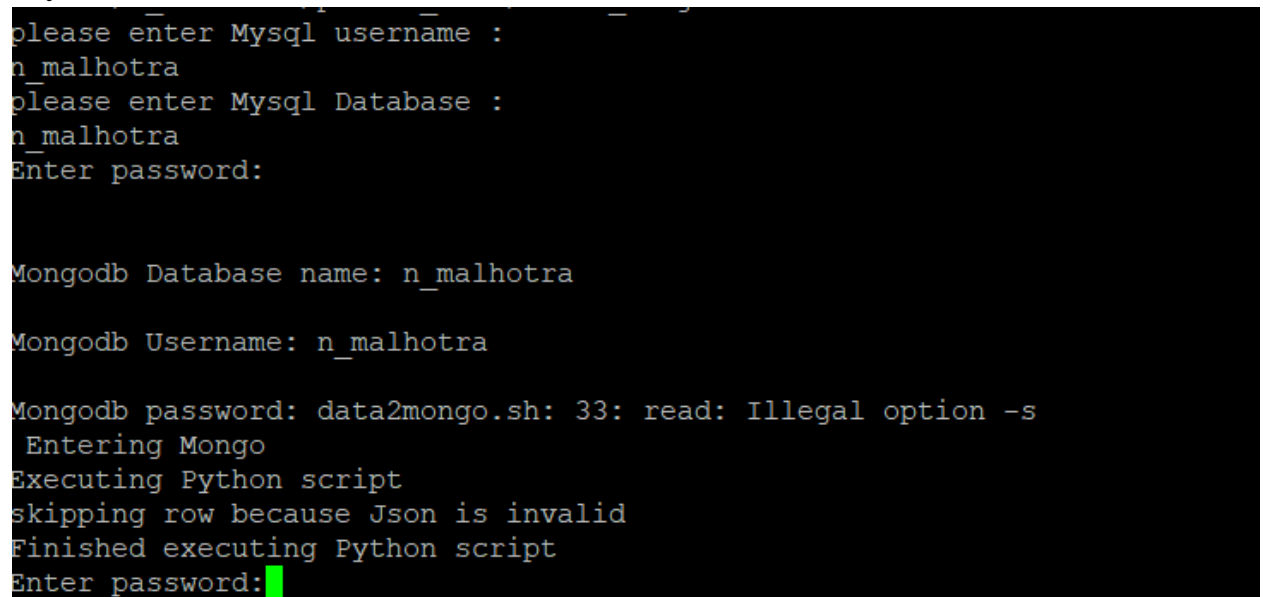
1. Validate the number of arguments passed to the script while running. If its not equal to one then display message, containing the expected output, and exit.
2. Connect to MYSQL database, by asking the user their credentials and database name.
3. Run ‘existing_tables.sql’ and ‘new_tables.sql’ files on the DB, to execute and create a fresh schema.
4. Run ‘cleanJSON.py’ python script. This script will contain all the logic to clean the articles data, and break it into 4 different JSON files, which will be loaded to Mongo DB.
5. Connect to MongoDB, by taking credentials from the user.
6. Add the data to mongo collections- ‘articles’, ‘Authors’, ‘magazines’, ‘author_articles’.
7. Delete the JSON files to clean up the directory.



```
n_malhotra@dev:~/public_html/Final_Project$ sh data2mongo.sh /home/student_2019_winter/n_malhotra/public_html/Final_Project
please enter Mysql username :
```

Figure 4.2.2.1 - Screenshot depicts passing file location as parameter

The above Screenshot depicts executing of .sh or Bash files executes in the command line, prompting user to enter enter the username used for connecting to MySQL.



```
please enter Mysql username :
n_malhotra
please enter Mysql Database :
n_malhotra
Enter password:

Mongodb Database name: n_malhotra

Mongodb Username: n_malhotra

Mongodb password: data2mongo.sh: 33: read: Illegal option -s
Entering Mongo
Executing Python script
skipping row because Json is invalid
Finished executing Python script
Enter password:
```

Figure 4.2.2.2 - Screenshot depicts bash script execution

The code snippet depicts execution of Bash script with elicitation of main points. The mongo script implicitly calls Python script which cleans data as defined and shown in later section.

```
skipping row because Json is invalid
Finished executing Python script
Enter password:

2019-04-14T15:45:46.362-0300    connected to: localhost
2019-04-14T15:45:46.362-0300    dropping: n_malhotra.articles
2019-04-14T15:45:49.335-0300    [#####.] n_malhotra.articles 1
8.4MB/18.9MB (97.4%)
2019-04-14T15:45:49.418-0300    [#####] n_malhotra.articles 1
8.9MB/18.9MB (100.0%)
2019-04-14T15:45:49.418-0300    imported 78441 documents
Enter password: █
```

Figure 4.2.2.3 - Screenshot depicts total rows inserted in collections.

This is the final command screen which depicts that the database (MongoDB) has imported 78441 documents.

4.2.3. Cleaning Data using Python script (‘cleanJSON.py’)-

Import library ‘json’

1. Initialize variables.
2. Open the files in which output json data is to be inserted.
3. Create a loop to read one JSON file at a time, and insert it into python dictionary element (type of python data structure, similar to json).
4. Initialize empty dictionaries for ‘articles’, ‘magazine’, ‘Authors’, ‘author_articles’, ‘volume’.
5. **Logic for ‘articles’ collection:**
 - Insert ‘title.ftext’ inside the ‘title’ property of articles dictionary.
 - If exception occurs, then check if the incoming property is string or list.
 - Read data accordingly, and add to articles.
 - Repeat for ‘pages’, ‘volume’, ‘year’, ‘author’ properties

Logic for ‘author’ collection:

- The authors property of articles.json file may contain a json containing ‘ftext’ or an array of jsons containing ‘ftext’.
- We first read the properties, and check if the author is a json/dictionary(in python), or an array (list in python).
- The data extraction logic is based on this condition, and the appropriate extraction of author names is done.
- This is now passed to ‘set’ object of python, which will remove all the duplicates in author.

- The name is split on basis of first ' ' (space character), and divided to first name and last name.
- If there is no last name then it will be inserted null (' ')

Logic for 'Magazine', 'Volume' Collections:

- The Magazine and Volume collections shouldn't contain duplicates. For this we again use python 'sets'.
- The data is inserted in dictionary as in above cases, and exported to json file.

7. Load the data from the output dictionary files to temporary output files.

8. Close all the files.

```
cleanJSON.py  X
1  #The script is called by data2mongo bash script
2  #reads article.json file row-by-row and put the data into articles,Authors, Magazines json files
3  #These json files are consistant, and in single structure.
4  #These are loaded in the corresponding mongo collections
5
6
7  import sys
8  import json
9  # fname = raw_input()
10 fname = "articles.json"
11 out_file = "jsonToMongoCleanArticles.json"
12 out_author = "jsonToMongoAuthors.json"
13 out_magazine = "jsonToMongoMagazine.json"
14 out_author_article = "jsonToMongoAuthorArticles.json"
15 out_volume = "jsonToMongoVolume.json"
16
17 #opening the files
18 try:
19     file = open(fname, "r")
20 except IOError:
21     print "\nError opening file!"
22     sys.exit()
23
24 try:
25     file_out_article = open(out_file, "w+")
26 except IOError:
27     print "\nError opening file!"
28     sys.exit()
29
30 try:
31     file_out_author = open(out_author, "w+")
32 except IOError:
33     print "\nError opening file!"
34     sys.exit()
35
36 try:
37     file_out_magazine = open(out_magazine, "w+")
38 except IOError:
39     print "\nError opening file!"
40     sys.exit()
```

Figure 4.2.3.1 - Importing JSON libraries and file creation.

In the above Screenshot, we create four variables which denote that we will create 4 collections holding JSON documents. We have have opened various files with read and write attribute and surrounded that code with exception handling blocks.

```
except IOError:
    print "\nError opening file!"
    sys.exit()

try:
    file_out_magazine = open(out_magazine, "w+")
except IOError:
    print "\nError opening file!"
    sys.exit()

try:
    file_out_authorArticle = open(out_author_article, "w+")
except IOError:
    print "\nError opening file!"
    sys.exit()

#defining variables
valid = True
autharticlevalid = True

dataout = {}
authorout = {}
magazineout = {}
volumeout = {}
authortemp = set()
magazinetemp = set()
authorarticle = {}
authorVolumeSet = set()
authorarticletext = ""
index = 0
magId = 1
```

Figure 4.2.3.2 - Creating Python dictionaries each storing respective JSON objects.

The above Screenshot depicts 4 instance creations of Python Dictionaries. Dictionaries map properties to value. Dictionaries were used in order to create JSON collections which will be used by respective tables.

```

# reading articles.json line-by-line
for i in file.readlines():
    index = index+1
    try:
        data = json.loads(i)
    except ValueError:
        print ("skipping row because json is invalid")
        valid = False
    if valid == True:
#adding title property to json
        try:
            dataout['title'] = data['title']['ftext']
            dataout['id'] = index
        except KeyError:
            try:
                if type(data['title']['i']).__name__ == "str":
                    dataout['title'] = data['title']['i']['ftext']
                    dataout['id'] = index
                    #if autharticlevalid == True:
                    authorarticle['title'] = data['title']['i']['ftext']
                    #authorarticletext = data['title']['i']['ftext']
                if type(data['title']['i']).__name__ == "list":
                    dataout['title'] = data['title']['i'][0]['ftext']
                    dataout['id'] = index
                    #if autharticlevalid == True:
                    authorarticle['title'] = data['title']['i'][0]['ftext']
                    #authorarticletext = data['title']['i'][0]['ftext']
            except:
                dataout['title'] = ""
                dataout['id'] = index
                autharticlevalid = False

```

Figure 4.2.3.3 -We fetch each JSON document from the collection and Validate its title property.

In this code snippet, we parse and process each document. Checks are placed to ensure we receive a document, define the 'title' property and after checking the datatype i.e. it is list, JSON or string, we follow appropriate method to insert data.

```

#adding pages property to json
try:
    dataout['pages'] = data['pages']['ftext']
except:
    dataout['pages'] = " "
# except NameError:
#     dataout['pages'] = " "
#adding volume property to json
try:
    dataout['volume'] = data['volume']['ftext']
    #
    authorVolumeDict['volume_id'] = data['volume']['ftext']
except KeyError:
    dataout['volume'] = data['volume']['i']['ftext']
    #
    authorVolumeDict['volume_id'] = data['volume']['i']['ftext']
#adding year property to json
try:
    dataout['year'] = data['year']['ftext']
    #
    authorVolumeDict['year'] = data['year']['ftext']
except:
    print(json.dumps(data))
# adding data to magazine collection
try:
    dataout['magazine'] = data['journal']['ftext']
    magazinetemp.add(data['journal']['ftext'])
except:
    try:
        dataout['magazine'] = data['booktitle']['ftext']
        magazinetemp.add(data['booktitle']['ftext'])
    except:
        print(json.dumps(data, indent=4))

```

Figure 4.2.3.4 - Depicts Validation of pages, volume and year property

We insert data and create properties for Volume, pages and year with their respective values from articles.JSON collection. Validations are placed to ensure uniqueness of data.

```

] #adding author property to json
] try:
]     if type(data['author']).__name__ == "dict":
]         #dataout['author'] = []
]         dataout['author'] = (data['author']['ftext'])
]         #if autharticlevalid == True:
]         authorarticle['author'] = data['author']['ftext']
]         authorarticle['articleId'] = index
]         file_out_authorArticle.write(json.dumps(authorarticle) + "\n")
]         authortemp.add(data['author']['ftext'])
]     if type(data['author']).__name__ == "list":
]         dataout['author'] = []
]         for author in data['author']:
]             dataout['author'].append(author['ftext'])
]             authortemp.add(author['ftext'])
]             #if autharticlevalid == True:
]             authorarticle['title'] = dataout['title']
]             authorarticle['author'] = author['ftext']
]             authorarticle['articleId'] = index
]             file_out_authorArticle.write(json.dumps(authorarticle) + "\n")
] # Handling exceptions
] except:
]     try:
]         if type(data['editor']).__name__ == "dict":
]             #dataout['author'] = []
]             dataout['author'] = (data['editor']['ftext'])
]             authortemp.add(data['editor']['ftext'])
]             #if autharticlevalid == True:
]             authorarticle['author'] = data['editor']['ftext']
]             authorarticle['articleId'] = index
]             file_out_authorArticle.write(json.dumps(authorarticle) + "\n")
]         if type(data['editor']).__name__ == "list":
]             dataout['author'] = []
]             for author in data['editor']:
]                 dataout['author'].append(author['ftext'])
]                 authortemp.add(author['ftext'])

```

Figure 4.2.3.5 - The code snippet depicts addition of properties to prepare final authorArticle.JSON and associated error handling

The above code snippet depicts a data type check as the author names are passed as JSON objects in some documents whereas lists in other. Each block has its own logic of inserting data into new Authors. Json. Also, it was observed that some author names are passes as editor names rather than author and hence we have programmed a similar block checking code to extract editor data.

```

        for author in data['editor']:
            dataout['author'].append(author['ftext'])
            authortemp.add(author['ftext'])
            #if autharticlevalid == True:
            authorarticle['title'] = dataout['title']
            authorarticle['author'] = author['ftext']
            authorarticle['articleId'] = index
            file_out_authorArticle.write(json.dumps(authorarticle) + "\n")

        except:
            dataout['author'] = []
            #authortemp.add('')
            autharticlevalid = False

        # print(json.dumps(dataout, indent=4))
        file_out_article.write(json.dumps(dataout) + "\n")
    # authorVolumeSet.add(authorVolumeDict)
    valid = True
    autharticlevalid = True
    # adding unique author names to Authors collection
    for author in authortemp:
        authorout['fname'] = author.split(' ',1)[0]
        try:
            authorout['lname'] = author.split(' ', 1)[1]
        except:
            authorout['lname'] = ''
            authorout['email'] = ''
        file_out_author.write(json.dumps(authorout) + "\n")

    for magazine in magazinetemp:
        magazineout['name'] = magazine
        file_out_magazine.write(json.dumps(magazineout) + "\n")

```

Figure 4.2.3.6 - Creating a unique Authors collection.

In this code screenshot, we create a 'set' type variable in order to store all the Author names. The set element is chosen to ensure that no author name appears twice, as authors can write multiple articles hence to insert unique author details only a set was formed.


```
file.close()
file_out_article.close()
file_out_magazine.close()
file_out_author.close()
file_out_authorArticle.close()
#file_out_authorArticle1.close()
```

Figure 4.2.3.7 - Closing all files. The control passes back to bash file.

In the above screenshots, we close connection to all the files and terminate the Python script.

4.2.4. Transferring data from Mongo to MySql ‘mongo2sql.sh’ Bash script:

1. Validate the number of arguments passed to the script while running. If its not equal to one then display message, containing the expected output, and exit.
2. Connect to Mongo DB using credentials provided by the user.
3. Export data to mongo collections -
 - ‘Authors to ‘authors.csv’
 - ‘Magazines’ to ‘magazines.csv’
 - ‘Articles’ to ‘articles.csv’
 - ‘Author_articles’ to ‘author_articles.csv’
 - ‘Articles’ to ‘volume.csv’
4. Connect to MYSQL database, and add data to respective tables.
5. Delete the intermediate files, for clean directory.

```

n_malhotra@dev:~/public_html/Final_Project$ sh mongo2sql.sh /home/student_2019_w
inter/n_malhotra/public_html/Final_Project

Mongodb Database name: n_malhotra

Mongodb Username: n_malhotra

Mongodb password: █

```

Figure 4.2.4.1 - Passing file to Bash script .

The screenshot depicts that a parameter is passed to the mongo collection. This parameter is the file location which in succession displays the username and user password.

```

2019-04-14T16:30:46.332-0300    connected to: localhost
2019-04-14T16:30:46.335-0300    exported 124 records
Importing Magazine Data from Mongo
2019-04-14T16:30:46.374-0300    connected to: localhost
2019-04-14T16:30:47.374-0300    [#####.....] n_malhotra.Authors
48000/119271 (40.2%)
2019-04-14T16:30:48.263-0300    [#####] n_malhotra.Authors
119271/119271 (100.0%)
2019-04-14T16:30:48.263-0300    exported 119271 records
Importing Magazine Data from Mongo
2019-04-14T16:30:48.309-0300    connected to: localhost
2019-04-14T16:30:49.309-0300    [#####.....] n_malhotra.articles
24000/78441 (30.6%)
2019-04-14T16:30:50.309-0300    [#####..] n_malhotra.articles
72000/78441 (91.8%)
2019-04-14T16:30:50.360-0300    [#####] n_malhotra.articles
78441/78441 (100.0%)
2019-04-14T16:30:50.360-0300    exported 78441 records
Importing Author articles Data from Mongo
2019-04-14T16:30:50.415-0300    connected to: localhost
2019-04-14T16:30:51.414-0300    [#####.....] n_malhotra.author_ar
ticles 56000/206608 (27.1%)
2019-04-14T16:30:52.414-0300    [#####.....] n_malhotra.author_ar
ticles 112000/206608 (54.2%)
2019-04-14T16:30:53.414-0300    [#####....] n_malhotra.author_ar
ticles 176000/206608 (85.2%)
2019-04-14T16:30:53.728-0300    [#####] n_malhotra.author_ar
ticles 206608/206608 (100.0%)
2019-04-14T16:30:53.739-0300    exported 206608 records
Importing Volume Data from Mongo
2019-04-14T16:30:53.792-0300    connected to: localhost
2019-04-14T16:30:54.792-0300    [#####.....] n_malhotra.articles
64000/78441 (81.6%)
2019-04-14T16:30:54.885-0300    [#####] n_malhotra.articles
78441/78441 (100.0%)
2019-04-14T16:30:54.885-0300    exported 78441 records
Entering MySQL
please enter Mysql username : █

```

Figure 4.2.4.2 - Importing Data in MySQL using MongoDB

```

2019-04-14T16:34:11.342-0300      [#####] n_malhotra.articles
78441/78441 (100.0%)
2019-04-14T16:34:11.342-0300      exported 78441 records
Entering MySQL
please enter Mysql username :
n_malhotra
please enter Mysql password :
A00430215
please enter Mysql Database :
n_malhotra
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql: [Warning] Using a password on the command line interface can be insecure.
n_malhotra@dev:~/public_html/Final_Project$ █

```

Figure 4.2.4.3 - Closing all files. The control passes back to bash file.

In the above Screenshots, we first import data into MySQL. Different CSV files were created which each fill data into respective database tables.

5. Conclusion

In this project, we achieved various objectives such as Visualization of an effective Database schema, transferring data between an RDBMS (MySQL) and NoSQL(MongoDB) via use of JSON files and scripting to automate the processes. We stored Halifax Science Libraries business information and implemented a Web application for the client.

6. Future Work

This project has scope for improvement. In terms of fast retrieval of data, we can index the tables which allow faster querying of data. Further, we plan to partition the data and create heap memory. We further aim to improve upon the data transference capability as data loading into tables slows the application. The application can be deployed on the server instead being run on local host.

7. Appendix A- [References]

- <https://stackoverflow.com/questions/4837673/how-to-execute-mongo-commands-through-shell-scripts> - Execute mongo using shell
- <https://www.youtube.com/watch?v=s1secsqSWLs>

- <https://realpython.com/python-json/>
- <https://stackoverflow.com/questions/17043860/python-dump-dict-to-json-file> - Python library to use json
- <https://stackoverflow.com/questions/14841461/display-an-error-message-under-certain-conditions-empty-list> - Display error
- <https://stackoverflow.com/questions/11772072/mvc-calling-a-view-from-a-different-controller> - How to call different controller in asp.net
- <https://stackoverflow.com/questions/8334493/get-table-names-using-select-statement-in-mysql> - How to get all table names in db
- <https://stackoverflow.com/questions/1665941/c-sharp-30-days-from-todays-date> - How to add 30 days logic for transaction
- <https://dotnetthoughts.net/how-to-make-string-contains-case-insensitive/> - How to compare string for fname and lname logic
- <https://forums.asp.net/t/1721626.aspx?Duplicate+Primary+Key+Validation> – Duplicate primary key validation

Appendix B- [SQL Scripts]

Transaction Table Script -

```
CREATE TABLE IF NOT EXISTS `project`.`transaction` (
  `trn_code` INT(11) NOT NULL,
  `date` VARCHAR(50) NULL DEFAULT NULL,
  `total_price` DECIMAL(10,2) NULL DEFAULT NULL,
  `cust_id` INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (`trn_code`),
  INDEX `fk_trn_cust` (`cust_id` ASC) VISIBLE,
```

```
CONSTRAINT `fk_trn_cust`  
  FOREIGN KEY (`cust_id`)  
  REFERENCES `project`.`customer` (`cust_id`))
```

Magazine Table creation

```
CREATE TABLE IF NOT EXISTS `project`.`magazine` (  
  `magazine_id` INT(11) NOT NULL,  
  `item_id` INT(11) NULL DEFAULT NULL,  
  `name` VARCHAR(50) NULL DEFAULT NULL,  
  PRIMARY KEY (`magazine_id`),  
  INDEX `fk_item_magazine` (`item_id` ASC) VISIBLE,  
  CONSTRAINT `fk_item_magazine`  
    FOREIGN KEY (`item_id`)  
    REFERENCES `project`.`library_item` (`item_id`))  
ENGINE = InnoDB
```

Volume tables creation

```
CREATE TABLE IF NOT EXISTS `project`.`volume` (  
  `volume_id` INT(11) NOT NULL,  
  `year` VARCHAR(5) NULL DEFAULT NULL,  
  `magazine_id` INT(11) NULL DEFAULT NULL,  
  PRIMARY KEY (`volume_id`),  
  INDEX `fk_volume_magazine` (`magazine_id` ASC) VISIBLE,  
  CONSTRAINT `fk_volume_magazine`  
    FOREIGN KEY (`magazine_id`)  
    REFERENCES `project`.`magazine` (`magazine_id`))
```

Articles table creation

```
CREATE TABLE IF NOT EXISTS `project`.`articles` (  
  `article_id` INT(11) NOT NULL,  
  `page` VARCHAR(50) NULL DEFAULT NULL,  
  `title` VARCHAR(50) NULL DEFAULT NULL,  
  `volume_id` INT(11) NULL DEFAULT NULL,  
  `year` VARCHAR(5) NULL DEFAULT NULL,  
  PRIMARY KEY (`article_id`),
```

```

INDEX `fk_volume_article` (`volume_id` ASC) VISIBLE,
CONSTRAINT `fk_volume_article`
  FOREIGN KEY (`volume_id`)
  REFERENCES `project`.`volume` (`volume_id`))

```

Author- Articles table creation

```

CREATE TABLE IF NOT EXISTS `project`.`author_articles` (
  `author_id` INT(11) NOT NULL,
  `article_id` INT(11) NOT NULL,
  INDEX `fk_authorArticle_article` (`article_id` ASC) VISIBLE,
  INDEX `fk_authorArticle_Author` (`author_id` ASC) VISIBLE,
  PRIMARY KEY (`author_id`, `article_id`),
  CONSTRAINT `fk_authorArticle_Author`
    FOREIGN KEY (`author_id`)
    REFERENCES `project`.`author` (`author_id`),
  CONSTRAINT `fk_authorArticle_article`
    FOREIGN KEY (`article_id`)
    REFERENCES `project`.`articles` (`article_id`))

```

Book table creation script -

```

CREATE TABLE IF NOT EXISTS `project`.`book` (
  `book_id` INT(11) NOT NULL,
  `item_id` INT(11) NULL DEFAULT NULL,
  `name` VARCHAR(50) NULL DEFAULT NULL,
  PRIMARY KEY (`book_id`),
  INDEX `fk_book_item` (`item_id` ASC) VISIBLE,
  CONSTRAINT `fk_book_item`
    FOREIGN KEY (`item_id`)
    REFERENCES `project`.`library_item` (`item_id`))

```

Book Author creation -

```

CREATE TABLE IF NOT EXISTS `project`.`book_author` (
  `book_id` INT(11) NOT NULL,
  `author_id` INT(11) NOT NULL,
  INDEX `fk_book_authorbook` (`book_id` ASC) VISIBLE,

```

```

INDEX `fk_authorbook_author` (`author_id` ASC) VISIBLE,
PRIMARY KEY (`author_id`, `book_id`),
CONSTRAINT `fk_authorbook_author`
  FOREIGN KEY (`author_id`)
  REFERENCES `project`.`author` (`author_id`),
CONSTRAINT `fk_book_authorbook`
  FOREIGN KEY (`book_id`)
  REFERENCES `project`.`book` (`book_id`))

```

Employee table creation -

```

CREATE TABLE IF NOT EXISTS `project`.`employee` (
  `SIN` VARCHAR(50) NOT NULL,
  `fname` VARCHAR(50) NULL DEFAULT NULL,
  `lname` VARCHAR(50) NULL DEFAULT NULL,
  `hourly_rate` DECIMAL(10,2) NULL DEFAULT NULL,
  PRIMARY KEY (`SIN`))

```

Monthly Expense table creation -

```

CREATE TABLE IF NOT EXISTS `project`.`monthly_expense` (
  `monthlyExpense_id` INT(11) NOT NULL,
  `year` VARCHAR(50) NULL DEFAULT NULL,
  `month` VARCHAR(50) NULL DEFAULT NULL,
  `cost_heat` DECIMAL(10,2) NULL DEFAULT NULL,
  `cost_water` DECIMAL(10,2) NULL DEFAULT NULL,
  `cost_electricity` DECIMAL(10,2) NULL DEFAULT NULL,
  `rent_id` INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (`monthlyExpense_id`),
  INDEX `fk_expense_rent` (`rent_id` ASC) VISIBLE,
  CONSTRAINT `fk_expense_rent`
    FOREIGN KEY (`rent_id`)
    REFERENCES `project`.`rent` (`rent_id`))

```

Employee- monthly expense table -

```

CREATE TABLE IF NOT EXISTS `project`.`mothlyexpense_employee` (
  `hours_worked` DECIMAL(10,2) NULL DEFAULT NULL,

```

```

`date` DATE NULL DEFAULT NULL,
`SIN` VARCHAR(50) NOT NULL,
`monthlyExpense_id` INT(11) NOT NULL,
PRIMARY KEY (`SIN`, `monthlyExpense_id`),
INDEX `fk_expense_monthly_expense` (`monthlyExpense_id` ASC) VISIBLE,
INDEX `fk_expense_monthly_employee` (`SIN` ASC) VISIBLE,
CONSTRAINT `fk_expense_monthly_employee`
  FOREIGN KEY (`SIN`)
    REFERENCES `project`.`employee` (`SIN`),
CONSTRAINT `fk_expense_monthly_expense`
  FOREIGN KEY (`monthlyExpense_id`)
    REFERENCES `project`.`monthly_expense` (`monthlyExpense_id`))

```

9. Appendix C- [Screenshots of Application]

- Home Page:

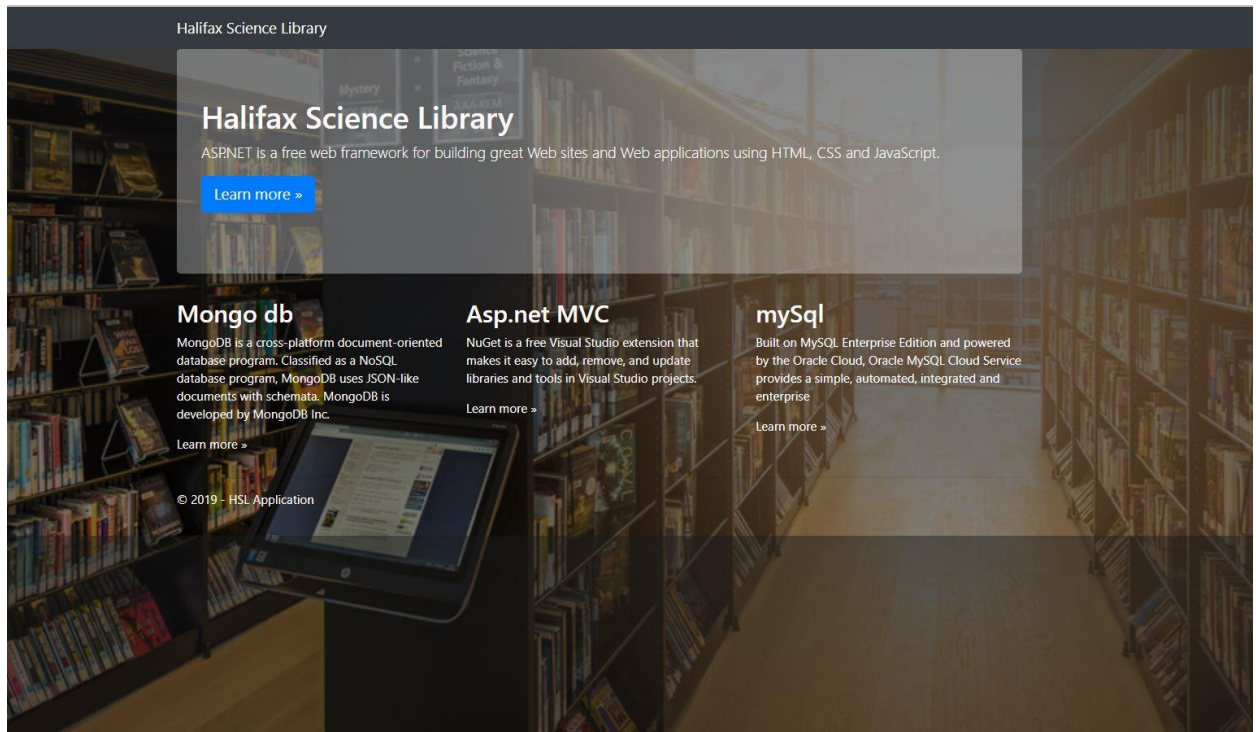


Figure 9.1 – This is the home page of our application Halifax science library

- Main page

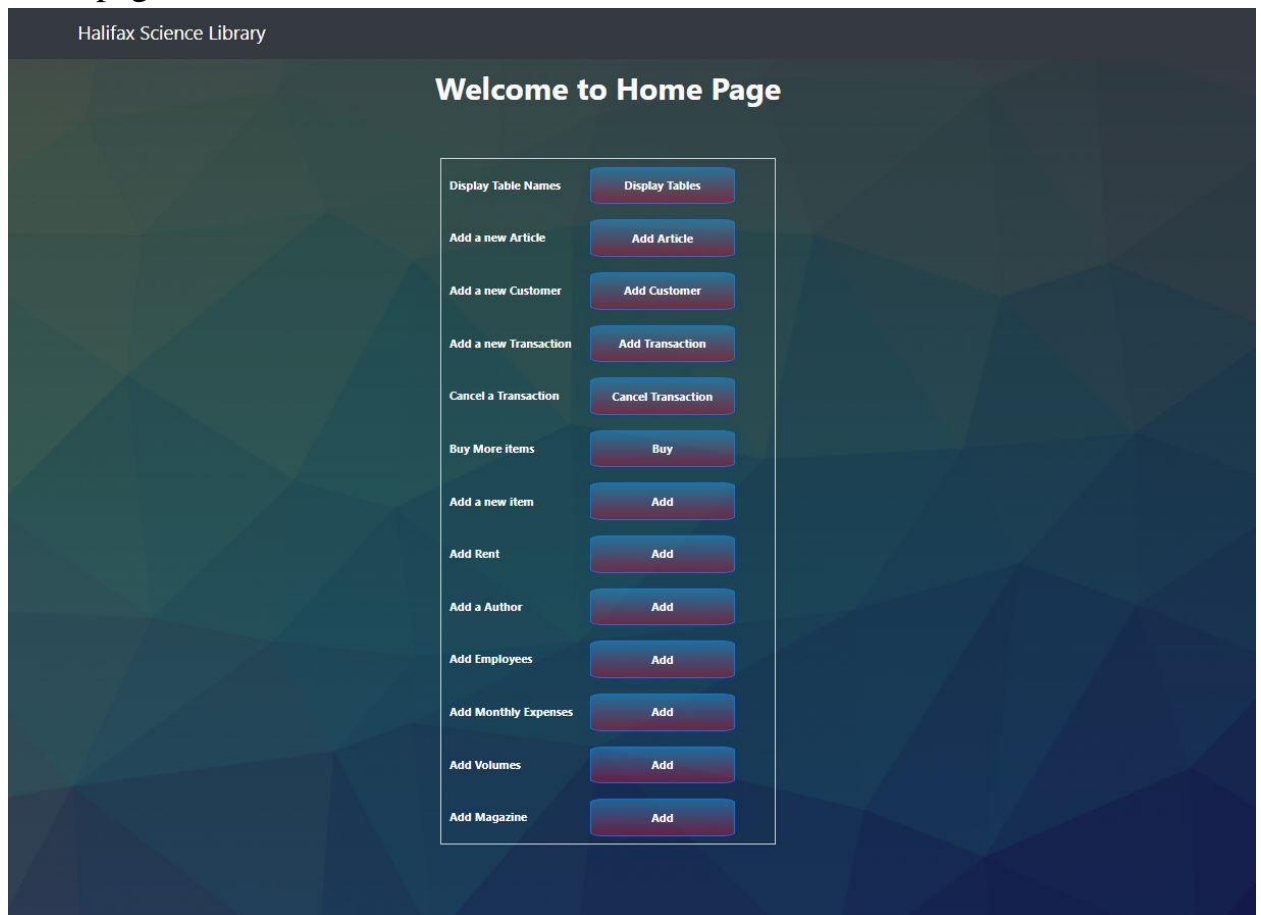


Figure 9.2 – This is the Menu page of our application Halifax science library showing all options

- Display Tables

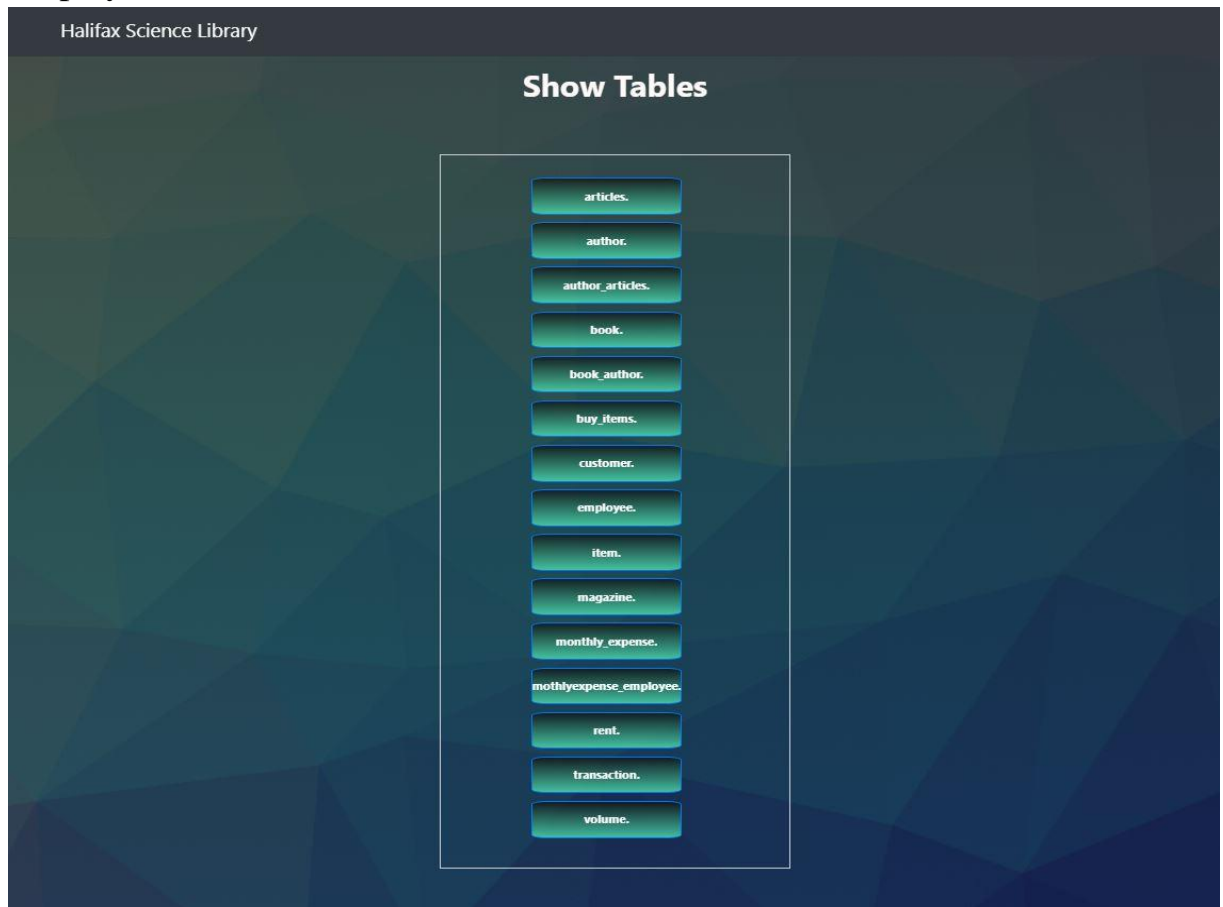


Figure 9.3 – This is the show all tables page of our application Halifax science library

- Display table content

Halifax Science Library

Articles Table

article_id	title	year	year	name	volume_id
1	Legend of Sahara	2002	2001		1
2	Legend of Sahara	2002	2001		1
3	Legend of Sahara	2002	2001		1
5	LetsCookIt	2003	2001	Theor. Comput. Sci.	1
10	Legend of Sahara	2012	2001	Lecture Notes in Computer Science	1

© 2019 - HSL Application

Figure 9.4 – This is the display table content of our application Halifax science library

- Create new record – Rent and Buy more items

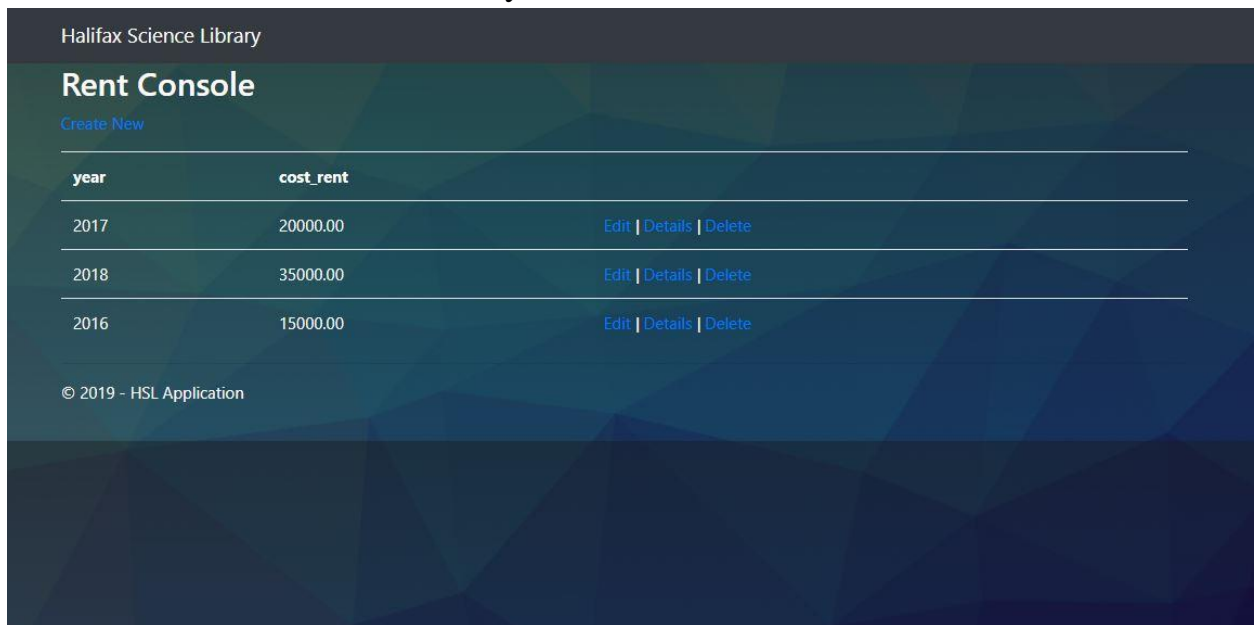


Figure 9.5 – This is the create rent page of our application Halifax science library

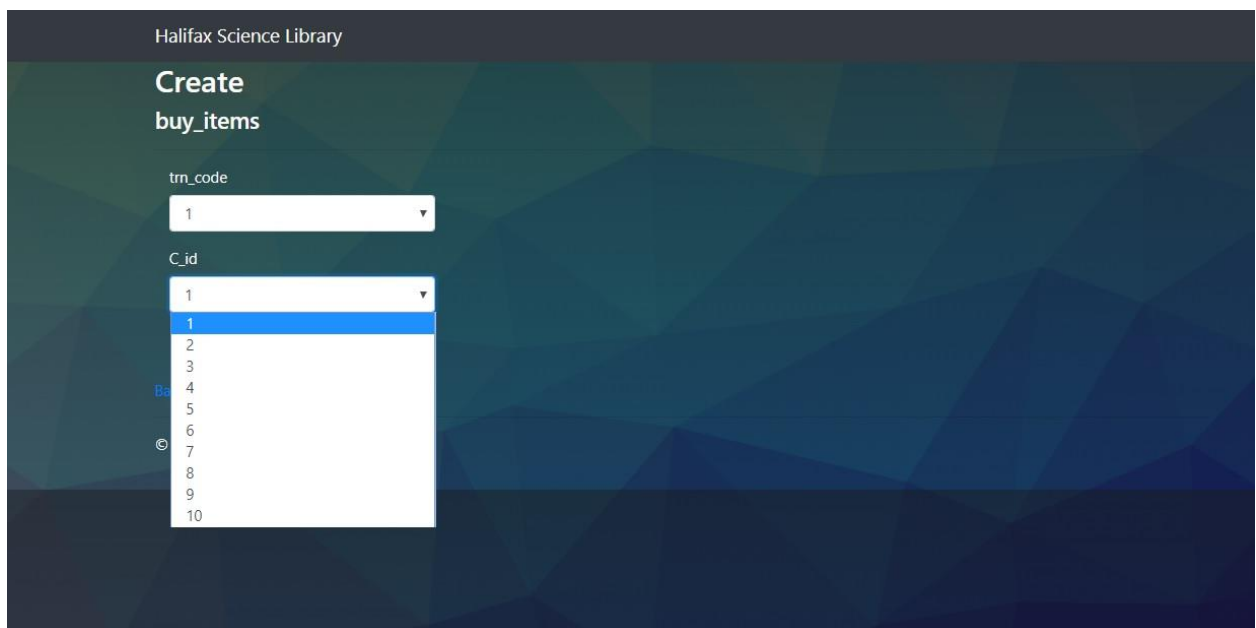


Figure 9.6 – This is the create buy items page of our application Halifax science library where a user can more than one item for same transaction code

- Creating new items

Halifax Science Library

Create item

price

Create

[Back to List](#)

© 2019 - HSL Application

Figure 9.7 – This is Create a new item page of our application Halifax science library

- Add new customer

Halifax Science Library

Customer Table

[Create New](#)

fname	lname	email	phone_num	discount_code	
shubham	Kumar	shubhamkumar@lfc@gmail.com	09090909333	0	Edit Details Delete
Sachit	Jain	SachitJain@gmail.com	09090909333	5	Edit Details Delete
shubham	Chumber	shubhamChumber@gmail.com	09090909333	3	Edit Details Delete
Vini	Mahajan	vini@gmail.com	9024121381	5	Edit Details Delete

© 2019 - HSL Application

Figure 9.8 – This is Create a new new customer page of our application Halifax science library

Halifax Science Library

Customer Table

[Create New](#)

Customer already exists with same First Name and Last Name !!

fname	lname	email	phone_num	discount_code	
shubham	Kumar	shubhamkumar@lfc@gmail.com	09090909333	0	Edit Details Delete
Sachit	Jain	SachitJain@gmail.com	09090909333	5	Edit Details Delete
shubham	Chumber	shubhamChumber@gmail.com	09090909333	3	Edit Details Delete
Vini	Mahajan	vini@gmail.com	9024121381	5	Edit Details Delete

© 2019 - HSL Application

Figure 9.8.1 – This is Create a new new customer page Validation of Fname and Lname of our application Halifax science library

- **ADD a new Author**

Halifax Science Library

Authors Console

[Create New](#)

lname	fname	email	
Abbad	Houda	houda.abbad@lycos.com	Edit Details Delete
Bannai	Hideo	bannai@inf.kyushu-u.ac.jp	Edit Details Delete
Beyazit	Mutlu	beyazit@adt.upb.de	Edit Details Delete
Blanchet-Sadri	Francine	blanchet@uncg.edu	Edit Details Delete
Brzozowski	Janusz	brzozo@uwaterloo.ca	Edit Details Delete
Campeanu	Cezar	cezar@sun11.math.upei.ca	Edit Details Delete
Caralp	Mathieu	mathieu.caralp@lif.univ-mrs.fr	Edit Details Delete
Caron	Pascal	pascal.caron@univ-rouen.fr	Edit Details Delete
Champarnaud	Jean-Marc	Jean-Marc.Champarnaud@univ-rouen.fr	Edit Details Delete
Chistikov	Dmitry	dch@mpi-sws.org	Edit Details Delete
Choffrut	Christian	Christian.Choffrut@liafa.jussieu.fr	Edit Details Delete
Crespi-Reghezzi	Stefano	stefano.crespireghizzi@polimi.it	Edit Details Delete
Debarbieux	Denis	denis.debarbieux@inria.fr	Edit Details Delete
Degano	Pierpaolo	degano@di.unipi.it	Edit Details Delete
Demaille	Akim	akim.demaille@gmail.com	Edit Details Delete

Figure 9.9 – This is Create a new Author page of our application Halifax science library

APPENDIX – D [data2mongo.sh]

```
#!/bin/bash
```

```
#the bash script expects folder path as argument
```

```
#It will first connect to mongo DB and export data to csv file from the  
collections
```

```
#Then it will connect to MySQL database and load data to tables from the  
generated csv files
```

```
if [ $# -ne 1 ]  
then  
    printf "\nUsage: \n$ $0 <folder/>\n\n"  
    exit 1  
fi
```

```
folder="$1"
```

```
# Connecting to MySQL Database
```

```
echo "please enter Mysql username : "
```

```
read user
```

```
echo "please enter Mysql Database : "
```

```
read sqlDB
```

```
#executing existing and new scripts
```

```
mysql -u "$user" -p"$pass" -D "$sqlDB" --local-infile -e "use $sqlDB;source  
existing_tables.sql;source new_tables.sql;"
```

```
echo ""
```

```
# change folder
```

```
cd $folder
```

```
#Connecting to MONGO DB
printf "\nMongodb Database name: "
read db
printf "\nMongodb Username: "
read userMongo
printf "\nMongodb password: "
read -s passMongo
echo " Entering Mongo "
cd $folder
```

```
#mongoimport -d "$db" -u "$userMongo" -p "$passMongo" -c "Authors" --
file "author_json.json" --drop
```

```
#Executing the python script for transforming data
echo "Executing Python script"
python cleanJSON.py
echo "Finished executing Python script"
```

```
#creating mongo collections form article.json
mongoimport -d "$db" -u "$userMongo" -p "$passMongo" -c "articles" --file
"jsonToMongoCleanArticles.json" --drop
mongoimport -d "$db" -u "$userMongo" -p "$passMongo" -c "Authors" --
file "jsonToMongoAuthors.json" --drop
mongoimport -d "$db" -u "$userMongo" -p "$passMongo" -c "magazines" --
file "jsonToMongoMagazine.json" --drop
mongoimport -d "$db" -u "$userMongo" -p "$passMongo" -c
"author_articles" --file "jsonToMongoAuthorArticles.json" --drop
```

```
#deleting the generated temp files
rm "jsonToMongoCleanArticles.json"
rm "jsonToMongoAuthors.json"
rm "jsonToMongoMagazine.json"
#rm "author_json.json"
```

```
#rm "jsonToMongoAuthorArticles.json"
```

Echo

APPENDIX – E [cleanJSON.py]

```
#The script is called by data2mongo bash script
#reads article.json file row-by-row and put the data into articles,Authors,
Magazines json files
#These json files are consistant, and in single structure.
#These are loaded in the corresponding mongo collections
```

```
import sys
import json
# fname = raw_input()
fname = "articles.json"
out_file = "jsonToMongoCleanArticles.json"
out_author = "jsonToMongoAuthors.json"
out_magazine = "jsonToMongoMagazine.json"
out_author_article = "jsonToMongoAuthorArticles.json"
out_volume = "jsonToMongoVolume.json"
```

```
#opening the files
```

```
try:
    file = open(fname, "r")
except IOError:
    print "\nError opening file!"
    sys.exit()
```

```
try:
    file_out_article = open(out_file, "w+")
except IOError:
    print "\nError opening file!"
    sys.exit()
```



```

try:
    file_out_author = open(out_author, "w+")
except IOError:
    print "\nError opening file!"
    sys.exit()

try:
    file_out_magazine = open(out_magazine, "w+")
except IOError:
    print "\nError opening file!"
    sys.exit()

try:
    file_out_authorArticle = open(out_author_article, "w+")
except IOError:
    print "\nError opening file!"
    sys.exit()

#defining variables
valid = True
autharticlevalid = True

dataout = {}
authorout = {}
magazineout = {}
volumeout = {}
authortemp = set()
magazinetemp = set()
authorarticle = {}
authorVolumeSet = set()
authorarticletext = ""
index = 0
magId = 1

```

```

# reading articles.json line-by-line
for i in file.readlines():
    index = index+1
    try:
        data = json.loads(i)
    except ValueError:
        print ("skipping row because Json is invalid")
        valid = False
    if valid == True:
#adding title property to json
        try:
            dataout['title'] = data['title']['ftext']
            dataout['id'] = index
        except KeyError:
            try:
                if type(data['title']['i']).__name__ == "str":
                    dataout['title'] = data['title']['i']['ftext']
                    dataout['id'] = index
                    #if autharticlevalid == True:
                    authorarticle['title'] = data['title']['i']['ftext']
                    #authorarticletext = data['title']['i']['ftext']
                if type(data['title']['i']).__name__ == "list":
                    dataout['title'] = data['title']['i'][0]['ftext']
                    dataout['id'] = index
                    #if autharticlevalid == True:
                    authorarticle['title'] = data['title']['i'][0]['ftext']
                    #authorarticletext = data['title']['i'][0]['ftext']
            except:
                dataout['title'] = ""
                dataout['id'] = index
                autharticlevalid = False
#adding pages property to json
        try:
            dataout['pages'] = data['pages']['ftext']
        except:

```

```

        dataout['pages'] = " "
    # except NameError:
    #     dataout['pages'] = " "
#adding volume property to json
    try:
        dataout['volume'] = data['volume']['ftext']
    #     authorVolumeDict['volume_id'] = data['volume']['ftext']
    except KeyError:
        dataout['volume'] = data['volume']['i']['ftext']
    #     authorVolumeDict['volume_id'] = data['volume']['i']['ftext']
#adding year property to json
    try:
        dataout['year'] = data['year']['ftext']
    #     authorVolumeDict['year'] = data['year']['ftext']
    except:
        print(json.dumps(data))
# adding data to magazine collection
    try:
        dataout['magazine'] = data['journal']['ftext']
        magazinetemp.add(data['journal']['ftext'])
    except:
        try:
            dataout['magazine'] = data['booktitle']['ftext']
            magazinetemp.add(data['booktitle']['ftext'])
        except:
            print(json.dumps(data, indent=4))
#adding author property to json
    try:
        if type(data['author']).__name__ == "dict":
            #dataout['author'] = []
            dataout['author'] = (data['author']['ftext'])
            #if autharticlevalid == True:
            authorarticle['author'] = data['author']['ftext']
            authorarticle['articleId'] = index
            file_out_authorArticle.write(json.dumps(authorarticle) + "\n")

```

```

        authortemp.add(data['author']['ftext'])
    if type(data['author']).__name__ == "list":
        dataout['author'] = []
        for author in data['author']:
            dataout['author'].append(author['ftext'])
            authortemp.add(author['ftext'])
            #if autharticlevalid == True:
            authorarticle['title'] = dataout['title']
            authorarticle['author'] = author['ftext']
            authorarticle['articleId'] = index
            file_out_authorArticle.write(json.dumps(authorarticle) + "\n")
# Handling exceptions
except:
    try:
        if type(data['editor']).__name__ == "dict":
            #dataout['author'] = []
            dataout['author'] =(data['editor']['ftext'])
            authortemp.add(data['editor']['ftext'])
            #if autharticlevalid == True:
            authorarticle['author'] = data['editor']['ftext']
            authorarticle['articleId'] = index
            file_out_authorArticle.write(json.dumps(authorarticle) + "\n")
        if type(data['editor']).__name__ == "list":
            dataout['author'] = []
            for author in data['editor']:
                dataout['author'].append(author['ftext'])
                authortemp.add(author['ftext'])
                #if autharticlevalid == True:
                authorarticle['title'] = dataout['title']
                authorarticle['author'] = author['ftext']
                authorarticle['articleId'] = index
                file_out_authorArticle.write(json.dumps(authorarticle) +
"\n")

    except:

```

```

        dataout['author'] = []
        #authorTemp.add("")
        autharticlevalid = False
        # print(json.dumps(dataout, indent=4))
        file_out_article.write(json.dumps(dataout) + "\n")
        # authorVolumeSet.add(authorVolumeDict)
        valid = True
        autharticlevalid = True
# adding unique author names to Authors collection
for author in authorTemp:
    authorout['fname'] = author.split(' ',1)[0]
    try:
        authorout['lname'] = author.split(' ', 1)[1]
    except:
        authorout['lname'] = ""
    authorout['email'] = ""
    file_out_author.write(json.dumps(authorout) + "\n")

for magazine in magazinTemp:
    magazineout['name'] = magazine
    file_out_magazine.write(json.dumps(magazineout) + "\n")

file.close()
file_out_article.close()
file_out_magazine.close()
file_out_author.close()
file_out_authorArticle.close()
#file_out_authorArticle1.close()

```

APPENDIX – F [mongo2sql.sh]

```
#!/bin/bash
```

```
#the bash script expects folder path as argument
#It will first connect to mongo DB and export data to csv file from the
collections
#Then it will connect to MySQL database and load data to tables from the
generated csv files
```

```
#Checking the number of arguments. Only folder name is expected
if [ $# -ne 1 ]
then
    printf "\nUsage: \n\$ $0 <folder/>\n\n"
    exit 1
fi
```

```
folder="$1"
```

```
#Changing the folder
cd $folder
```

```
#Connecting to Mongo Database
```

```
printf "\nMongodb Database name: "
read db
printf "\nMongodb Username: "
read userMongo
printf "\nMongodb password: "
read passMongo
echo " Entering Mongo "
```

```
#Importing data from Mongo collections to csv files
```

```
echo "Importing Magazine Data from Mongo"
mongoexport -d "$db" -u "$userMongo" -p "$passMongo" -c "magazines" --
type=csv --fields name --out magazines.csv
echo "Importing Magazine Data from Mongo"
```

```
mongoexport -d "$db" -u "$userMongo" -p "$passMongo" -c "Authors" --  
type=csv --fields fname,lname,email --out authors.csv
```

```
echo "Importing Magazine Data from Mongo"
```

```
mongoexport -d "$db" -u "$userMongo" -p "$passMongo" -c "articles" --  
type=csv --fields id,pages,title,volume,year --out articles.csv
```

```
echo "Importing Author_articles Data from Mongo"
```

```
mongoexport -d "$db" -u "$userMongo" -p "$passMongo" -c  
"author_articles" --type=csv --fields author,title --out author_articles.csv
```

```
echo "Importing Volume Data from Mongo"
```

```
mongoexport -d "$db" -u "$userMongo" -p "$passMongo" -c "articles" --  
type=csv --fields volume,year --out volume.csv
```

```
echo " Entering MySQL "
```

```
#Connecting to MYSQL
```

```
echo "please enter Mysql username : "
```

```
read user
```

```
echo "please enter Mysql password : "
```

```
read pass
```

```
echo "please enter Mysql Database : "
```

```
read sqlDB
```

```
#Loading data to MYSQL
```

```
mysql -u "$user" -p"$pass" -D "$sqlDB" --local-infile -e "use  
n_malhotra;LOAD DATA LOCAL INFILE 'magazines.csv' into table  
MAGAZINE FIELDS TERMINATED BY ',' lines terminated by '\n'  
IGNORE 1 ROWS (name)"
```

```
mysql -u "$user" -p"$pass" -D "$sqlDB" --local-infile -e "use  
n_malhotra;LOAD DATA LOCAL INFILE 'authors.csv' into table  
AUTHOR FIELDS TERMINATED BY ',' lines terminated by '\n' IGNORE  
1 ROWS (fname,lname,email)"
```

```
mysql -u "$user" -p"$pass" -D "$sqlDB" --local-infile -e "use
n_malhotra;LOAD DATA LOCAL INFILE 'articles.csv' into table
ARTICLES FIELDS TERMINATED BY ',' lines terminated by '\n'
IGNORE 1 ROWS (article_id,page,title,volume_id,year)"
mysql -u "$user" -p"$pass" -D "$sqlDB" --local-infile -e "use
n_malhotra;LOAD DATA LOCAL INFILE 'author_articles.csv' into table
AUTHOR_ARTICLES FIELDS TERMINATED BY ',' lines terminated by
'\n' IGNORE 1 ROWS (author_name,article_title)"
mysql -u "$user" -p"$pass" -D "$sqlDB" --local-infile -e "use
n_malhotra;LOAD DATA LOCAL INFILE 'volume.csv' into table
VOLUME FIELDS TERMINATED BY ',' lines terminated by '\n' IGNORE
1 ROWS (volume_id,year)"
```

#article table update in new sql

```
#mysql -u "$n_malhotra" -p"A00430215" -D "n_malhotra" --local-infile -e
"use n_malhotra;LOAD DATA LOCAL INFILE 'articles.csv' into table
ARTICLES FIELDS TERMINATED BY ',' lines terminated by '\n'
IGNORE 1 ROWS (page,title,volume_id,year)"
```

```
rm "$folder/magazines.csv"
rm "$folder/authors.csv"
rm "$folder/articles.csv"
rm "$folder/author_articles.csv"
rm "$folder/volume.csv"
```