Extraction Section:

**MariaDB**

MariaDB has a couple different ways of extracting data from the database itself into CSV. By far the most popular way, and the way that’s the easiest to find within the MariaDB documentation is through the usage of “SELECT INTO OUTFILE.” Through this method you would execute it similarly to how a query is executed through command line of MariaDB. Through this method you would have to appropriately select the columns that you would be interested in gather data from followed by “INTO OUTFILE” with the desired path for the file. Through this method we can terminate fields by the required comma and terminate lines with the required field to match the CSV format. The following is an example of the format of the command to execute to output a table to csv format of the Employee\_Information Table.

SELECT Employee\_ID, DOB, DOJ, Department\_ID

INTO OUTFILE ‘Employee\_Information.csv’

FIELDS TERMINATED BY ‘,’

LINES TERMINATED BY ‘\n’

FROM Employee\_Information;

Second popular way through command line structure is through the client itself by executing the desired queries to fetch the data and then by using output redirection to file within the command.

mysql -u your\_username -p -e “SELECT Employee\_ID, DOB, DOJ, Department\_ID FROM Employee\_Information” MRU > Employee\_Information.csv

-p will prompt to enter password for the database and the -e will execute the command that follows and the second line is file output redirection from the screen/command line to a file.

Final way is very similar to the second but it can be done through the mysqldump command that is offered by MariaDB. Due to the way the data is dump this requires piping the output from the mysqldump into a SED to parse it into the required csv format but would result in a CSV output. The following can be executed to achieve this result

Mysqldump -u your\_username -p MRU Employee\_Information

| sed ‘s/\t/”,”/g;s/\n/\\n/g’ > Employee\_Information.csv

Note: assumptions made in the commands is that the table is Employee\_Information and that there may be additional fields in the table not outputted and that database name is MRU

**MongoDB**

Like MariaDB, MongoDB has a couple different ways of extracting data from the database into CSV as well. By far the most popular method or recommended method from the MongoDB documentation is through the use of the “mongoexport” command line tool that comes with MongoDB already. This tool allows you to export the data from your MongoDB to either JSON or CSV or any other format supported. The basic command line structure to achieve this would look like the following

mongoexport --host MRU --db MRU --collection Employee\_Information --type=csv --out Employee\_Information.csv --fields Employee\_ID,DOB,DOJ,Department\_ID

If you are looking for a more graphical way of extracting the data from MongoDB, MongoDB does provide a GUI called MongoDB Compass that provides various tools for working with MongoDB databases. One of these tools allows you to export query results to CSV format. To use this method first you would need to open MongoDB Compass and make a connection to your MongoDB server itself, navigate to the appropriate collection you wish to export data from, execute any required queries to filter the data as needed. Once done there is a button labeled “Export” button and select “Export Collection” these may be represented by small downward arrows depending on the version and selected CSV as the export option.

MongoDB can also be exported using pythons’ panda’s library that we have been using throughout the semester to export to a CSV file but could be seen as the most involved method in achieving the extraction process. This process can be achieved through the following python code

import pandas as pd

from pymongo import MongoClient

client = MongoClient('mongodb://MRU:27017/')

db = client['MRU']

collection = db['Employee\_Information']

cursor = collection.find({}, {'Employee\_ID': 1, 'DOB': 1, 'DOJ': 1, 'Department\_ID': 1, '\_id': 0})

df = pd.DataFrame(list(cursor))

df.to\_csv('Employee\_Information.csv', index=False)

Transformation:

For the most part, Transformation was relatively straightforward. Most fields you can simply check that they all are filled in and directly check for any out of range values. This will directly solve most issues, especially for our purposes in regression.

Trouble arises in searching for uniqueness among tables, especially the larger ones such as the performance table. I simply used a concatenation of any relevant PK elements and fed that into a dictionary, as searching an array is far too slow to be used in this manner, it also allowed me to keep track of the indexes (at least in the CSV file) for future reference concerning the DB admin cleaning their actual database. FK checks were made, but found nothing of concern, so the DBMS seems to be doing it’s job in retaining consistency there.

In terms of Completeness, We had pinged and deleted those lines with missing values, however there is a likely chance the data may be missing a few performances. With our dataset, our sample should still be reflective, but this blind area is worth noting.

For Validity, we had checked implausible or impossible values for Effort and Results, and deleted them for the sake of our analysis. It was presumed that Dates were stored in MM/DD/YYYY format, but there is a chance that most of the database is wrong. Regardless, this only brought up a few employee birthdates as incorrect.

In terms of Consistency, there were no FK deletion errors found. Good Job.

For uniqueness, there was surprisingly no duplicate assignments for any of the students. For the other tables however, there are several different duplicate entries, at least with primary keys which was what I checked on. Each duplicate pings to the previously found entry of that data using Python Dictionaries. In this DB there was no triplicate information, but in a different database, one would have to check for transient duplicate references when viewing the reports.

All of this is summarized in each of the Report csv files, where the index refers to the position of that item in the csv extracts given to us.