**Design:**

We have explored two probable ways in design the database using the Extraction, Transform and Load (ETL) process – one way is to create three different collections one for each data file; other approach is to create all the data fields into one flat structure that would allow the queries and thereby the user experience better. Added to that the intention is to store the aggregated values as well. Though each approach has its (dis) advantages we intended to design as a typical Data Warehouse approach that would facilitate analyzing the data better in more like Big Data approach as opposed to a Relation Data Model approach.

We have created the database called **MovieLens** with three collection in it. The three collections are **movies, tags and ratings**. The collection and the data fields in each collection as follows.

Movies – (MovieID, Title, Genre)

Tags – (UserId, MovieId, Title, Tag, Timestamp)

Ratings – (UserId, MovieId, Title, Genres, Rating, Timestamp).

Since the queries to be answered would require an equivalent of JOIN in a typical Relational data model we have enriched the Tags data with Title and created the tags collection; similarly the ratings collection has been enriched Genres.

**Implementation:**

Since the team is more comfortable with the Java programming language over Python, Java has been our choice to implement the ETL process. We have used FileIOStream processing with BufferedReader objects to enhance the performance of the ETL process one for each collection (or each data file). Using other Java classes by importing necessary Java Packages of core Java processing as well as that of MongoDB and instantiated the objects. Using a traditional loop-thru process we processed all the three files and created and loaded the data into MonogDB. Though the intention was to have aggregated data fields in the collections the system configuration limitation has restricted us not to include any aggregation while Loading the collections; otherwise we would have implemented by transferring the overhead of aggregations into one-time Loading process.

**Queries:**

The collection helped to us to answers queries in the best possible way and we could answer them correctly. We designed the collection keeping in view the three additional queries. Also we came up with the three additional queries to analyze or understand the customer or Users interests like what kind movies users Rated (watched), what kind of comments or tags that users made as if we were the owners of a video or movie library. In fact we also thought about finding if any patters between Users and the Genres they rated and relations among various Genres; but that would demand more recursive or advanced functions and hence could not implement.

Overall it has been a good experience to start working on a NoSQL databases (away from traditional RDBMS and the more we work the more interesting it has been so far.

**Execution:**

The MongoLoad.java file expects the Filepath which contains all the 3 files(movies.dat,tags.dat and ratings.dat) as the first argument while executing. The Mongo DB Server should be running on the localhost and port 27017(default values).

The following jars should be added to the classpath before executing it ;

mongodb-driver-3.0.4.jar, bson-3.0.4.jar, mongodb-driver-core-3.0.4.jar

Note : Recommended Argument to the JVM : -Xmx1200m