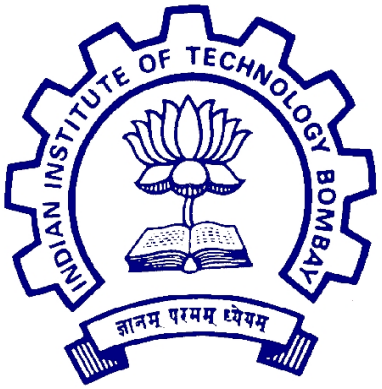
Data Analytics for IITBombayX (based on OpenEDX insights)

Summer Internship 2015

Submitted in fulfilment of internship Project



Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Powai, Mumbai

By
The Data Analytics Group

Project Mentor and Guide: Sukla Nag
Principal investigator: Dr. D. B. Phatak

July 2, 2015

THE TEAM

1. Ankit Kumar
   Indian Institute of Technology, Patna

2. Anurag Das
   Indian Institute of Information Technology, Allahabad

3. Arinjoy Basak
   Indian Institute of Engineering Science and Technology, Shibpur

4. Ashwani Pandey
   National Institute of Technology, Silchar

5. Dewang Palav
   Visveshwaraiya National Institute of Technology, Nagpur

6. Ekta Sharma
   Universal College of Engineering, Vasai

7. Jay Bothra
   Sardar Vallabhbhai National Institute of Technology, Surat

8. Geddam Nageshbabu
   Kakinada Institute of Engineering and Technology, Kakinada

9. Piridi Janaki
   Kakinada Institute of Engineering and Technology-W, Korangi

10. Sagar Agarwal
    Motilal Nehru National Institute of Technology, Allahabad

11. Zarana Parekh
    Dhirubhai Ambani Institute of Information and Communication Technolgy, Gandhinagar

Summer Internship 2015
Project Approval Certificate

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

The project entitled **Data Analytics For IITBombayX(Based on OPENedX insights)** submitted by Mr. Ankit Kumar, Mr. Anurag Das, Mr. Arinjoy Basak, Mr. Ashwini Pandey, Mr. Dewang Palav, Ms. Ekta Sharma, Mr. Jay Bothra, Mr. Geddam Nageshbabu, Ms. Piridi Janaki, Mr. Sagar Agarwal and Ms. Zarana Parekh, is approved for Summer Internship 2015 programme from 9th May 2015 to 3rd July 2015, at The Department of Computer Science and Engineering, IIT Bombay on successful completion of their project.

Prof. D.B. Phatak	Mrs. Sukla Nag
Dept of CSE, IITB	Dept of CSE, IITB
Principal Investigator	Mentor & Project Incharge

Place : IIT Bombay, Mumbai
Date : July 2, 2015

Declaration

I declare that this written submission represents my ideas in my own words and where other's ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Ankit Kumar
IIT Patna

Anurag Das
IIIT Allahabad

Arinjoy Basak
IIEST Shibpur

Ashwani Pandey
NIT Silchar

Dewang Palav
VNIT Nagpur

Ekta Sharma
UEC Vasai

Jay Bothra
SVNIT Surat

Geddam Nageshbabu
KIET Kakinada

Piridi Janaki
KIET-W Korangi

Sagar Agarwal
MNNIT Allahabad

Zarana Parekh
DA-IICT Gandhinagar

Abstract

OpenEdX source has been made available to the open source community as different modules such as LMS(Learning Management System),CMS(Content Management System),edx-ora(Open Response Assessment) etc.In this project we have tried to successfully integrate and run these different modules. OpenEdX hosts online university-level courses in a wide range of disciplines to a worldwide audience at no charge, and conducts research into learning.OpenEdX-platform generates tremendous amounts of data related to student, instructor, staff and course.Analysis of OpenEdX data enables us to answer the fundamental questions about learning nature of students. For Data analysis of the OpenedX data, the generated data must be extracted from different sources, cleaned and properly structured and finally stored in the Hadoop and Spark cluster because of its Big Data nature. This report elaborates and summarizes the working of a Data Analytics group created at IIT Bombay for managing such voluminous data to obtain structured results from unstructured and unarranged data sources. Additionally it further enhances user experience by providing a set of tools for visualising this data so that both researchers and professors can infer vital results with respect to EdX data. The Adaptation of OpenedX Insight has been optimized using the latest Big Data technologies to create both an efficient and fault tolerant architecture.

Acknowledgement

We, the summer interns of the team Data Analytics for IIT BombayX based on OpenEdX Insights, are overwhelmed in all humbleness and gratefulness to acknowledge our deep gratitude to all those who have helped us put our ideas to perfection and have assigned tasks, well above the level of simplicity and into something concrete and unique.We, whole heartedly thank Prof. D.B. Phatak for having faith in us, selecting us to be a part of his valuable project and for constantly motivating us to do better. We are very thankful to our project incharge,Mrs. Sukla Nag for her valuable suggestions.She was always there to show us the right track when needed help. With help of her brilliant guidance and encouragement, we all were able to complete our tasks properly and were up to the mark in all the tasks assigned. During the process, we got a chance to see the stronger side of our technical and non-technical aspects and also strengthen our concepts. Here by, we gladly consider ourselves to be the most fortunate batch of interns. Last but not the least, we whole heartedly thank all our other colleagues working in different projects under Prof. D.B Phatak for helping us evolve better with their critical advice.

List of Figures

Contents

Chapter 1

Introduction

1.1 Educational Data Mining In Big Data

Educational Data Mining refers to techniques, tools, and research designed for automatically extracting meaning from large repositories of data generated by or related to peoples learning activities in educational settings. Quite often, this data is extensive, fine-grained, and precise.At a high level, the field seeks to develop and improve methods for exploring this data, which often has multiple levels of meaningful hierarchy, in order to discover new insights about how people learn in the context of such settings.In doing so, EDM has contributed to theories of learning investigated by researchers in educational psychology and the learning sciences.

1.2 Improving Trend in EDM

As numerous articles in both the academic and popular press have pointed out, the ability of eDX to generate a tremendous amount of data opens up considerable opportunities for educational research. edX and Coursera, which together claim almost four and a half million enrollees, have developed platforms that track students every click as they use instructional resources ,complete assessments ,and engage in social interactions. These data have the potential to help researchers identify ,at a finer resolution than ever before ,what contributes to students learning and what hampers their success. eDX generates large amounts of data which needs to be processed. Why collect and store large amounts of data if you cant analyze it in full context? Or if you have to wait hours or days to get results? The idea is that if we know everything about a student, well be better able to help that student and future students who fit a similar profile.

1.3 What is OpenEdX?

OpenEdX is the open-source release of the EDX platform developed by the non-profit organization founded by Harvard and MIT.As OpenEdX is open source, other universities and educational providers can use it freely to support their own online learning initiatives.Universities and other organizations using OpenEdX also control the license for their content. They may release or redistribute that content in a variety of forms such as original, revised or remixed,to multiple audiences without special permissions from a

platform owner. This flexibility gives educators the freedom to experiment and to greatly expand the reach of their instructional materials.

1.4 What is OpenEdX Insights?

OpenEdX Insights makes information about courses available to course team members, which consists of Course Staff or Instructor role.OpenEdX Insights provides these course team members with data about student activity, background, and performance throughout the course.OpenEdX Insights can help one to monitor how students are doing, and validate the choices one made in designing there course. OpenEdX Insights includes a brief description for each reported value.It can also help to re-evaluate choices and inform efforts to improve the course and the experience of the learners. Putting the data provided by OpenEdX Insights to work involves the following points:

1. Evaluating reported data against the expectations and hypotheses.
2. Understanding the context of the course run,which includes the environmental factors and choices that make each run unique.
3. Deciding whether action is called for.
4. Selecting the action to take, and when.

1.5 Important results obtained

A data analytics system is required to setup a workflow to take different types of data generated as a source pipeline the data and process accordingly according to its type to ensure that it is ready to be transformed and loaded.The logs generated are unstructured so they need to be structured and cleaned so that they can be queried for useful information. It is also needed for analysing the large amounts of data that is generated and thus derive some useful results from it. Data visualisation is a modern branch of descriptive statistics. It involves the creation and study of the visual representation of data, meaning information that has been abstracted in some schematic form, including attributes or variables for the units of information.To provide with a better understanding of the results obtained from the analysis of theeDX data,data visualization can be used. The second aim of this project is to make the data analytics system suitable for edX data such that it is able to provide visualizations for different results.

Chapter 2

Architecture of OpenEDX Insights

2.1 Introduction

This chapter pertains to the functioning and detailed explanation of the entire OpenEdX platform. The various components of the system are explained in detail with their specification and working.

2.2 Architecture

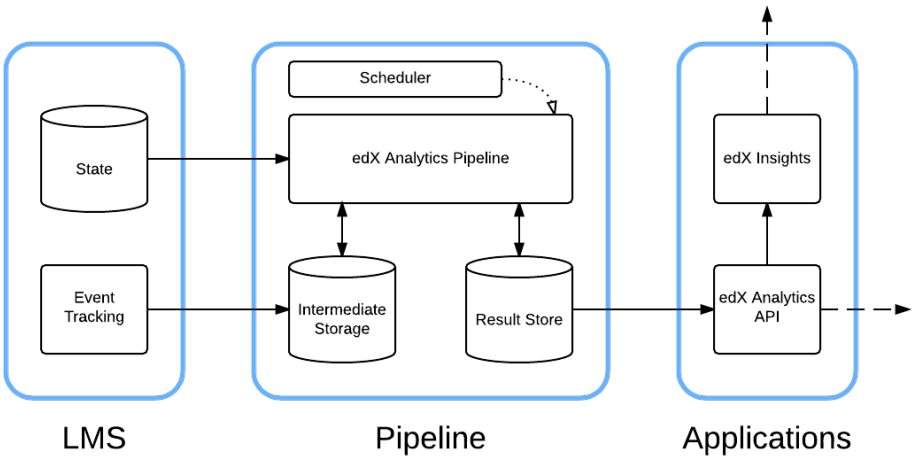


Figure 2.1: Insights Architecture

2.2.1 LMS

- The data taken as the summary data (state) stored in the MySQL database **edxapp** with the forum and discussion data stored in the mongoDB in the **modulestore** and `cs_comments_services` .

- All the user interactions with the system on millissecond basis are recorded by the system as event log record. These records are generated as json objects and stored in text files which are zipped, on daily basis. For their usage in the system they are stored on an Amazon S3 cloud from where they are fetched as and when required.

2.2.2 Pipeline

- The intermediate storage :
  - It takes data from zipped log files saved on hdfs on S3 and the summary data from SQL database and mongoDB.
- The edx analytics pipeline :
  - It uses SQOOP to transfer the appropriate SQL tables directly from the database to the pipeline.
  - Using the summary data and the log data performs map reduce jobs on Hadoop to generate the tables of the result store.
  - It reads the log files and finds the appropriate events , combines them with appropriate summary data by transferring it every time and does processing for a model. This transfer/scanning of log data takes place every time the batch process is run for populating the data into the respective models. Everytime the batch process is run log data is **scanned and cleaned** for every model
- Scheduler :
  - All the above tasks ranging from transferring log files from LMS to hdfs, to the map reduce job tasks are each separately written as luigi tasks. Luigi schedule these tasks in a batch process which is then run from time to time updating the result store .

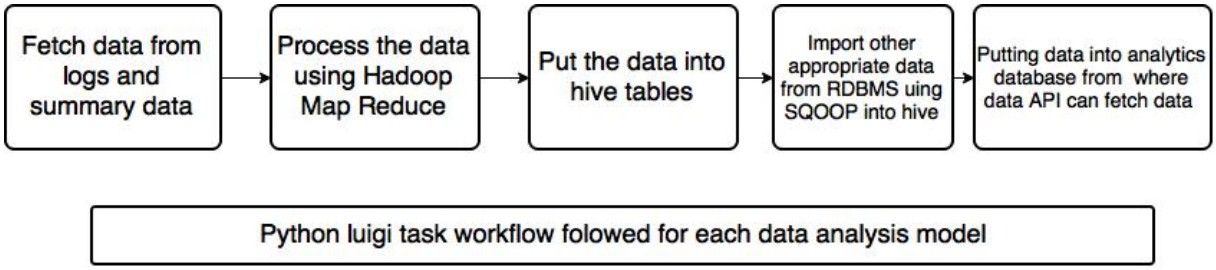


Figure 2.2: Python Luigi Task Workflow

- ResultStore :
  - It is the mySQL database with the name Analytics which is being populated with all the results which are being visualized directly on the dasboard (frontend) with the help of data API client through simple SQL queries on the database.

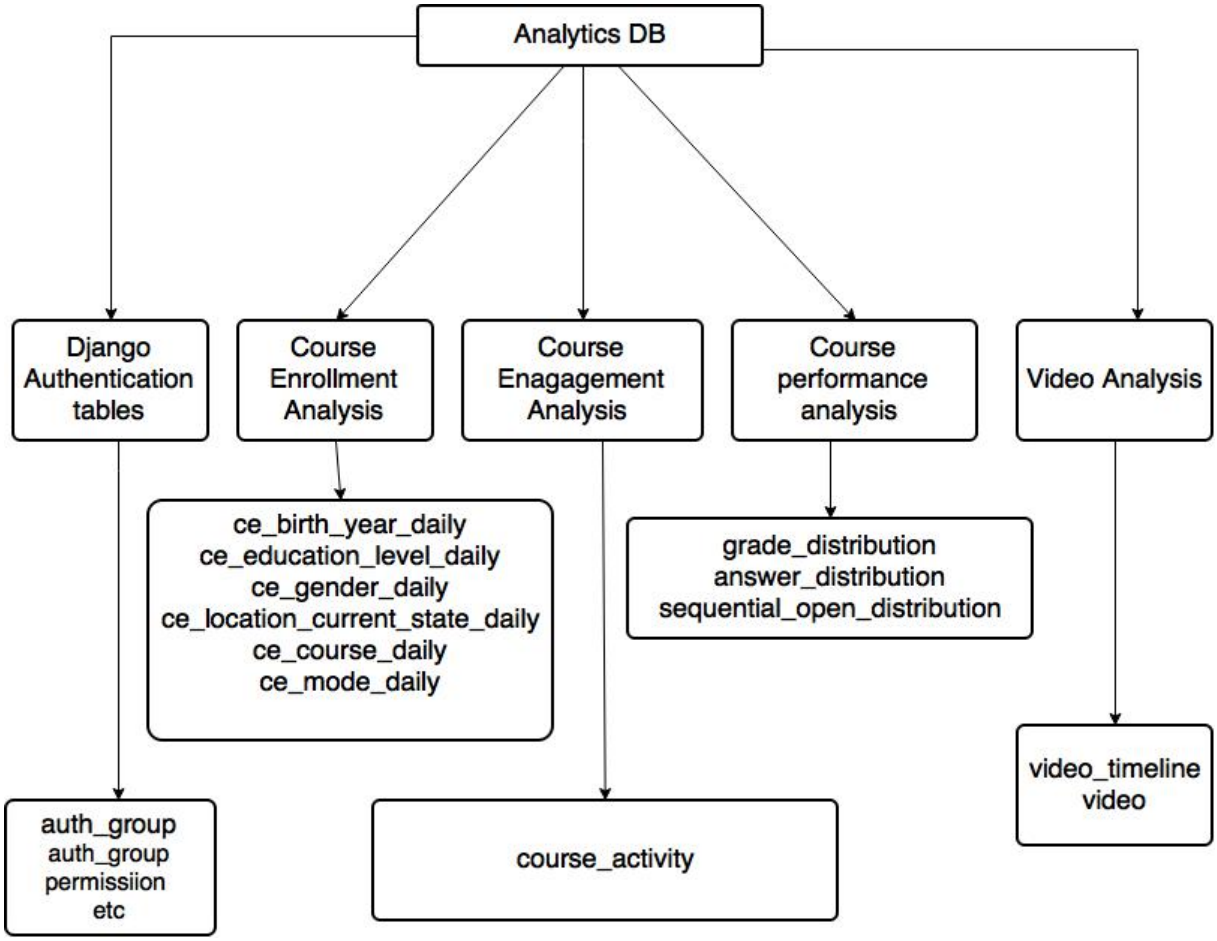


Figure 2.3: The Analytics Database

2.2.3 Applications(The Data API and the Dashboard)

- The Data API fetches data from the Analytics DB (Resultstore).
- It provides the fetched data to the Insights Dashboard to visualize depending on the request of the Insights Dashboard.

Chapter 3

Technologies Used

- **HTML** HyperText Markup Language, commonly referred to as HTML,is the language that describes the structure and the semantic content of a web document.HTML elements form the building blocks of all websites.
- **CSS** Cascading Style Sheets, most of the time abbreviated as CSS, is a stylesheet language used to describe the presentation of a document written in HTML or XML (including various XML languages like SVG or XHTML). CSS describes how the structured element must be rendered on screen, on paper, in speech, or on other media.
- **JavaScript** JavaScript is a powerful and popular language for programming on the web.It is a lightweight, dynamic, interpreted programming language. It is designed for creating network-centric applications. It is complimentary to and integrated with Java.
- **jQuery** It is a free and open source JavaScript library that is used by Web developers to navigate HTML documents, handle events, perform animations and add Ajax interactions to Web pages. It is designed to simplify the client-side scripting of HTML.
- **R** R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS.
  - **Why use R?** R is powerful software for interacting with data. With R you can create sophisticated graphs, you can carryout statistical analyses, and you can create and run simulations. R is also a programming language with an extensive set of built-in functions, so you can, with some experience, extend the language and write your own code to build your own statistical tools. Advanced users can even incorporate functions written in other languages, such as C, C++, and Fortran
  - **Why use R for introductory statistics?** There are several reasons that make R an excellent choice of statistical software for an introductory statistical course. First, R is free and available on the Web. You can use it on your home computers and are not tied to campus labs. Second, R is a powerful, widely-used software. The knowledge of R you gain during the course potentially translates to a marketable skill. You will learn to use a tool that has many practical uses outside the classroom. Third, even though it is not the simplest

statistical software, the basics are easy enough to master that learning to use R need not interfere overly much with learning the statistical concepts encountered in an introductory course.

plyr plyr is an R package that makes it simple to split data apart, do stuff to it, and mash it back together. This is a common data-manipulation step. Importantly, plyr makes it easy to control the input and output data format from a syntactically consistent set of functions.

RMySQL RmySQL is a Database Interface and MySQL driver for R. It implements DBI-compliant Interface to MySQL and MariaDB Databases.

googleVis is an R interface to Google Charts API, allowing users to create interactive charts based on data frames. Charts are displayed locally via the R HTTP help server. A modern browser with Internet connection is required and for some charts a Flash player. The data remains local and is not uploaded to Google.

- **Django** Django is an advanced Web framework written in Python that makes use of the model view controller (MVC) architectural pattern. Its key objective is to ease the development of complicated, database-driven websites.

- **Luigi** Luigi is a Python package that helps you build complex pipelines of batch jobs. It handles dependency resolution, workflow management, visualization, handling failures, command line integration, and much more.

- **Apache Spark** Spark is a fast and general cluster computing system for Big Data. It provides high-level APIs in Scala, Java, and Python, and an optimized engine that supports general computation graphs for data analysis. It also supports a rich set of higher-level tools including Spark SQL for SQL and DataFrames, MLlib for machine learning, GraphX for graph processing, and Spark Streaming for stream processing.

- **Apache Hadoop** Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

- **Apache Hive** The Apache Hive data warehouse software facilitates querying and managing large datasets residing in distributed storage. Hive provides a mechanism to project structure onto this data and query the data using a SQL-like language called HiveQL. At the same time this language also allows traditional map/reduce programmers to plug in their custom mappers and reducers when it is inconvenient or inefficient to express this logic in HiveQL.

- **Sqoop** Apache Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases.

- **Python** Python is a multiparadigm, general-purpose, interpreted, high-level programming language. Python allows programmers to use different programming styles to create simple or complex programs, get quicker results and write code almost as if speaking in a human language.

- **MySQL** MySQL is the most popular Open Source Relational SQL database management system. MySQL is one of the best RDBMS being used for developing web-based software applications. Both MySQL server and workbench were used.
- **Python Packages** The various python packages used are sasl, pyhs2, pymongo, pygoip,pydoop.hdfs,mysqlDB(mysql-connector-python)
- **Apache Tomcat Server** Apache Tomcat, often referred to as Tomcat, is an open-source web server and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements several Java EE specifications including Java Servlet, JavaServer Pages (JSP), Java EL, and WebSocket, and provides a "pure Java" HTTP web server environment for Java code to run in.

Chapter 4

Installation

This Chapter covers all the description and installation of all the tools required in this project. All the softwares and packages used are Open Source software and are freely available.

Figure 4.1: Technologies used

4.1 System Requirements

For Single Node : We have developed the whole system on the following hardware, and have used the following software.

4.1.1 Hardware Requirements

4.1.1.1 Hardware

- Dual core Intel Pentium compatible processor or multiprocessor-based computer with a 2Ghz or greater processor
- 64-bit system
- Network interface card

4.1.1.2 Disk Space

5 GB free disk space (minimum).
Requirements increase as data is gathered and stored in HDFS.

4.1.1.3 Memory

4 GB or more (recommended)

4.1.2 Software Requirements

- Operating System : Linux (Ubuntu 14.04 LTS)
- Web browser : Mozilla Firefox 28.0
- Text Editor : Vim
- Optional software : RStudio, edx-analytics-dashboard, edx-analytics,pipeline, edx-analytics-data-api

4.2 Hadoop 2.6.0

Apache Hadoop is an open-source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware.

Hadoop Distributed File System (HDFS) is a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster.

The Apache Hadoop framework is composed of the following modules:

- Hadoop Common : Contains libraries and utilities needed by other Hadoop modules
- Hadoop Distributed File System (HDFS) : A distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster.

- Hadoop YARN : A resource-management platform responsible for managing compute resources in clusters and using them for scheduling of usersâĂŹ applications.
- Hadoop MapReduce : A programming model for large scale data processing

4.2.1 Hadoop Stack

1. HDFS
   The Hadoop Distributed File System is the customized file system made for the Hadoop Ecosystem which supports large block sizes and coordinates storage between multiple DataNodes

2. MapReduce
   Programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster

3. Hive
   Hive allows SQL developers to write Hive Query Language (HQL) statements that are similar to standard SQL statements; now you should be aware that HQL is limited in the commands it understands

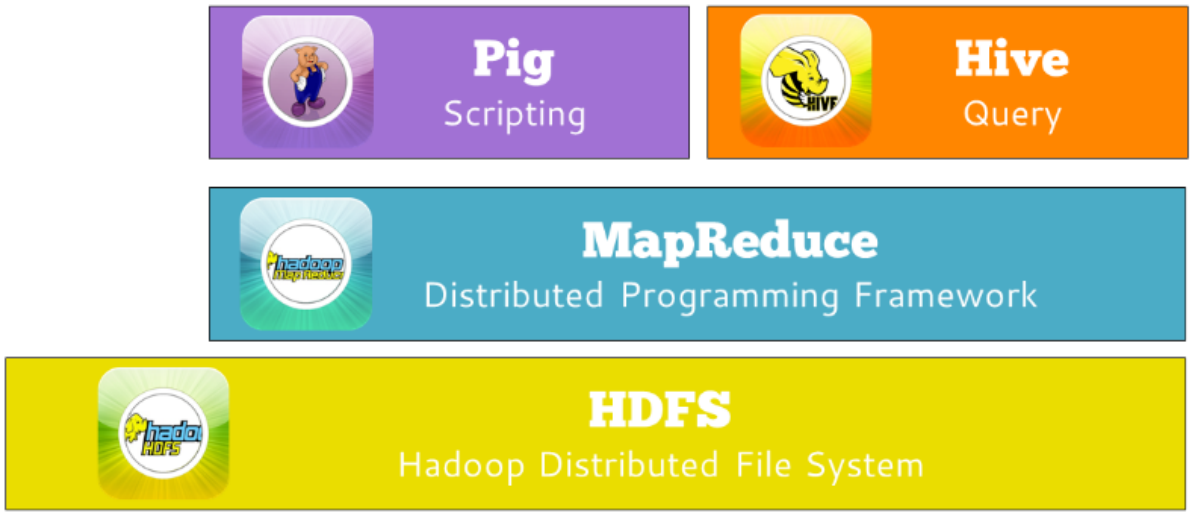


Figure 4.2: Technologies used

4.2.2 Prerequisites for Installation

1. Installing Oracle Java 8

   ```
   sudo apt-get install openjdk-8-jdk
   ```

2. Creating a Hadoop user for accessing HDFS and MapReduce

   ```
   sudo addgroup hadoop
   sudo adduser --ingroup hadoop hduser
   ```

3. Installing SSH

```
sudo apt-get install openssh-server
```

4. Configuring SSH

```
# First login with hduser (and from now use only hduser account for further steps
sudo su hduser
ssh-keygen -t rsa -P ""
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

5. Disabling IPv6 For getting your IPv6 disable in your Linux machine, you need to update /etc/sysctl.conf by adding following line of codes at end of the file,

```
# disable ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

4.2.3 Steps for Installation

Download latest Apache Hadoop source from Apache mirrors

```
## Locate to hadoop installation parent dir
cd /usr/local/

## Extract Hadoop source
sudo tar -xzvf hadoop-2.6.0.tar.gz

## Move hadoop-2.6.0 to hadoop folder
sudo mv hadoop-2.6.0 /usr/local/hadoop

## Assign ownership of this folder to Hadoop user
sudo chown hduser:hadoop -R /usr/local/hadoop

## Create Hadoop temp directories for Namenode and Datanode
sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode
sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode

## Again assign ownership of this Hadoop temp folder to Hadoop user
sudo chown hduser:hadoop -R /usr/local/hadoop_tmp/
```

4.2.4 User based configuration

1. Update Hadoop configuration files
   User profile : Update /.bashrc

```
## User profile : Update $HOME/.bashrc
sudo gedit .bashrc

## Update hduser configuration file by appending the
## following environment variables at the end of this file.

# -- HADOOP ENVIRONMENT VARIABLES START -- #
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
# -- HADOOP ENVIRONMENT VARIABLES END -- #
```

Configuration file : hadoop-env.sh

```
## To edit file, fire the below given command
/usr/local/hadoop/etc/hadoop$ sudo gedit hadoop-env.sh

## Update JAVA_HOME variable,
JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

Configuration file : core-site.xml

```
## To edit file, fire the below given command
/usr/local/hadoop/etc/hadoop$ sudo gedit core-site.xml

## Paste these lines into <configuration> tag
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
```

Configuration file : hdfs-site.xml

```
## To edit file, fire the below given command
/usr/local/hadoop/etc/hadoop$ sudo gedit hdfs-site.xml

## Paste these lines into <configuration> tag
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
```

```
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/usr/local/hadoop_tmp/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/usr/local/hadoop_tmp/hdfs/datanode</value>
</property>
```

Configuration file : yarn-site.xml

```
## To edit file, fire the below given command
/usr/local/hadoop/etc/hadoop$ sudo gedit yarn-site.xml

## Paste these lines into <configuration> tag
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>

##properties for memory management of yarn clusters
<property>
<name>yarn.nodemanager.resource.memory-mb</name>
<value>5200</value>
<description>Amount of physical memory, in mb that can be allocated for containe
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
</property>

<property>
<name>yarn.scheduler.minimum-allocation-mb</name>
<value>1500</value>
<description>Amount of physical memory, in mb that can be allocated for containe
<property>
```

Configuration file : mapred-site.xml

```
## Copy template of mapred-site.xml.template file
cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template  /usr/local/hadoop/etc/h

## To edit file, fire the below given command
```

```
/usr/local/hadoop/etc/hadoop$ sudo gedit mapred-site.xml

## Paste these lines into <configuration> tag
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
## The following properties can be changed depending on the memory requirements c
<property>
<name>mapreduce.map.memory.mb</name>
<value>2000</value>
</property>
<property>
<name>mapreduce.reduce.memory.mb</name>
<value>2000</value>
</property>
<property>
<name>yarn.app.mapreduce.am.resource.mb</name>
<value>4000</value>
<source>mapred-site.xml</source>
</property>
```

2. Format Namenode

```
hdfs namenode -format
```

4.2.5 Start Hadoop Cluster

1. Start all Hadoop daemons

   Start hdfs daemons

   ```
   /usr/local/hadoop$ start-dfs.sh
   ```

   Start MapReduce daemons:

   ```
   /usr/local/hadoop/start-yarn.sh
   ```

2. Track/Monitor/Verify

   ```
   ##Java Virtual Machine Process Status Tool
   jps
   ```

   The output should look like as shown in figure 4.3

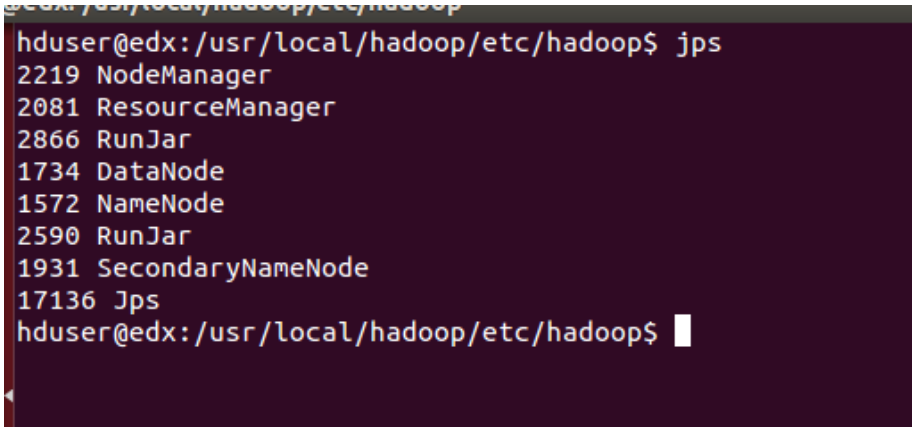


Figure 4.3: List of Hadoop services started

Another way to check is using netstat

```
netstat -plten | grep java
```

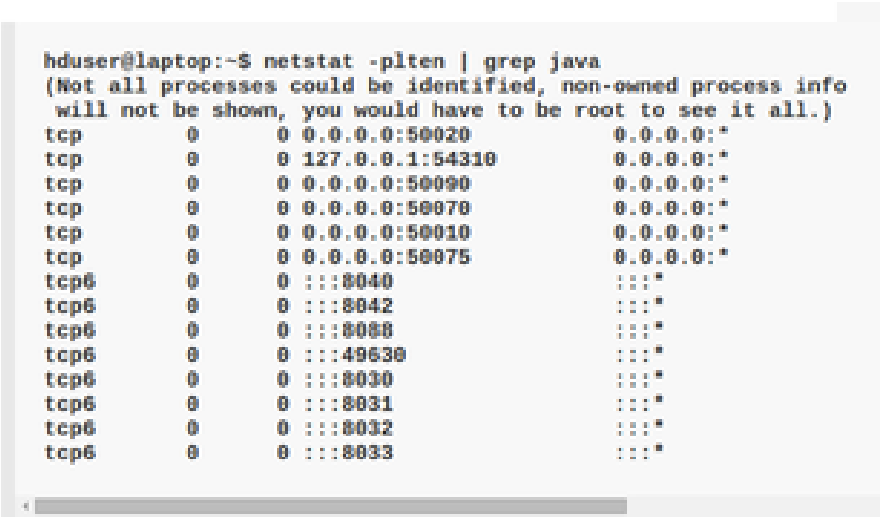


Figure 4.4: Output of Netstat Command

4.2.6 How Hadoop works

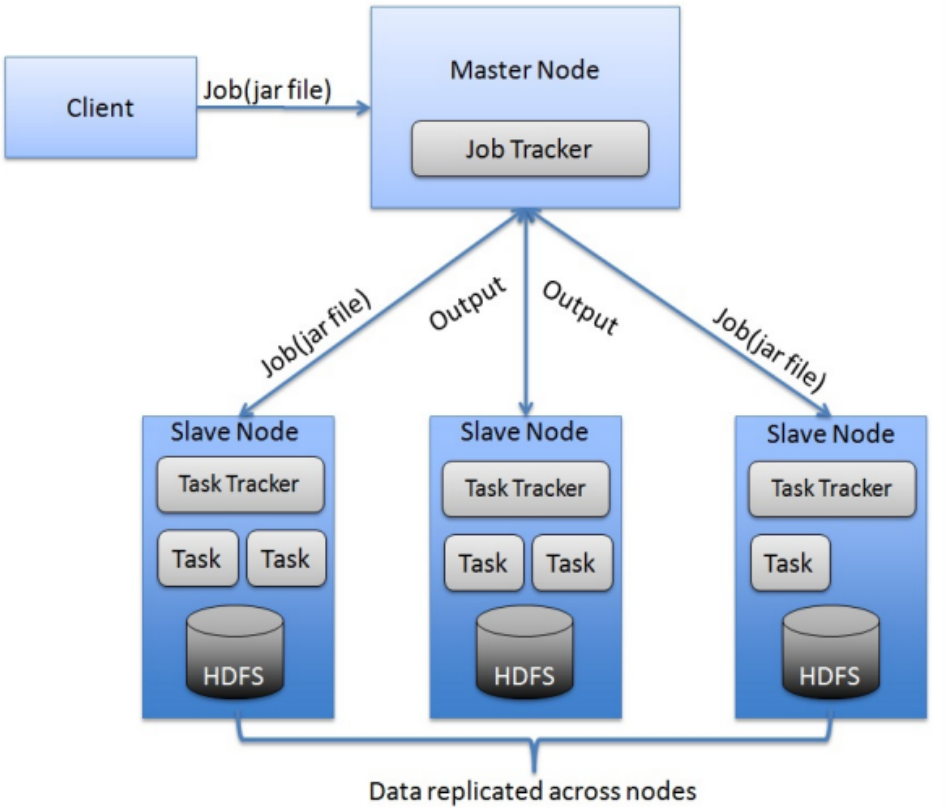


Figure 4.5: List of Hadoop services started

Hadoop and its various components fit together to ensure a fault-tolerant,durable and highly efficient model for storage and management of Big Data

(a) **Namenode**
Namenode is the node which stores the filesystem metadata i.e. which file maps to what block locations and which blocks are stored on which datanode. The namenode maintains two in-memory tables, one which maps the blocks to datanodes (one block maps to 3 datanodes for a replication value of 3) and a datanode to block number mapping. Whenever a datanode reports a disk corruption of a particular block, the first table gets updated and whenever a datanode is detected to be dead (because of a node/network failure) both the tables get updated.

(b) **Secondary Namenode**
The secondary namenode regularly connects to the primary namenode and keeps snapshotting the filesystem metadata into local/remote storage. It does so at a poor frequency and should not be heavily relied on.

(c) **Datanode**
The data node is where the actual data resides.
Points to Note:
1. All datanodes send a heartbeat message to the namenode every 3 seconds

to say that they are alive.
2. If the namenode does not receive a heartbeat from a particular data node for 10 minutes, then it considers that data node to be dead/out of service and initiates replication of blocks which were hosted on that data node to be hosted on another data node.
3. The data nodes can talk to each other to rebalance data, move and copy data around and keep the replication high.
4. When the datanode stores a block of information, it maintains a checksum for it as well.
5. The data nodes updates the namenode with the block information periodically and before updating verify the checksums.
6. If the checksum is incorrect for a particular block i.e. there is a disk level corruption for that block, it skips that block while reporting the block information to the namenode.
7. In this way, namenode is aware of the disk level corruption on that datanode and takes steps accordingly.

(d) **Node Manager**
This is a yarn daemon which runs on individual nodes and receive updated information on recource containers from their individual datanodes via background daemons. Different resources such as memory, cpu time, network bandwidth etc. are put into one unit called the Resource Container The Node manager in turn ensures fault tolerancy on the data nodes for any map reduce jobs

(e) **Resource Manager**
This is a yarn daemon which manages the allocation of resources to the different jobs apart from comprising a scheduler which just takes care of the scheduling jobs without worrying about any monitoring or status updates

4.2.7 Usage of Hadoop Commands

1. copyFromLocal

```
Usage:hadoop fs -copyFromLocal <local-file> <hadoop-fs-dir>
Details:Copies file from local file system to HDFS
```

2. copyToLocal

```
Usage:hadoop fs -copyToLocal <hadoop-fs-file> <local-dir>
Details:Copies file from local file system to HDFS
```

3. mkdir

```
Usage:hadoop fs -mkdir <hadoop-fs-dir>
Details:Create directory in the Hadoop Filesystem
```

4. ls

```
Usage:hadoop fs -ls <hadoop-fs-dir>
Details:List files inside a Hadoop Directory
```

5. put

   ```
   Usage:hadoop fs -put <localsrc> ... <HDFS_dest_Path>
   Details:Transfer file from local directory to hadoop file directory
   ```

6. chmod

   ```
   Usage: hadoop fs -chmod 777 <file path whose permission needs to be changed>
   Details:Change the permission of the file
   ```

7. rm -rf

   ```
   Usage: hadoop fs -rm -rf <folder path which is needed to be deleted>
   Details:deletes the folder and its contents in hadoop file system
   ```

   Note : Similarly other linux commands can also be used

4.2.8 Stopping Hadoop

The command used to stop hadoop is :

```
stop-dfs.sh
stop-yarn.sh
```

4.2.9 Hadoop Web Interfaces

4.2.9.1 Hadoop Web UI

Address : http://localhost:50070

This web UI is used to monitor the state of the cluster, namenodes, datanodes. It is also used for browsing hdfs directory and the log files.

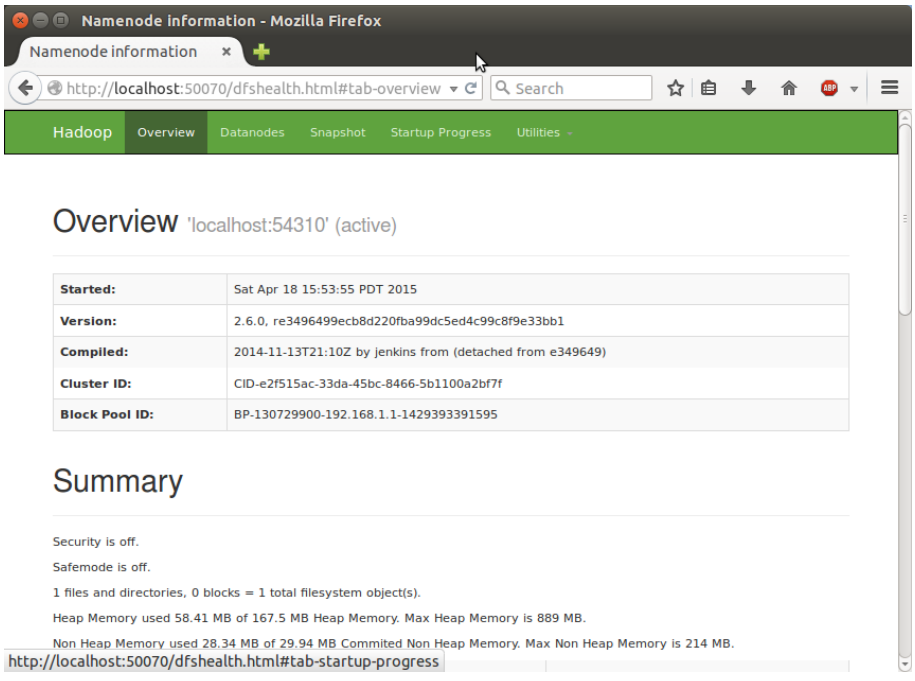


Figure 4.6: Web UI of Hadoop

4.2.9.2 Secondary NameNode Status Panel

Address : http://localhost:50090/status.jsp

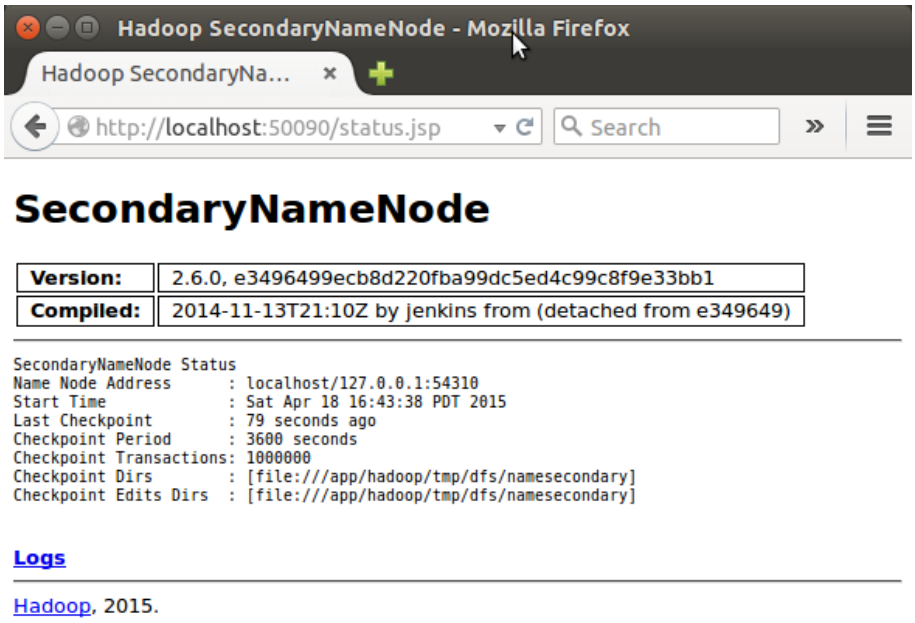


Figure 4.7: Hadoop Secondary NameNode

4.2.9.3 Hadoop Map Reduce Cluster UI

Address : http://localhost:8088

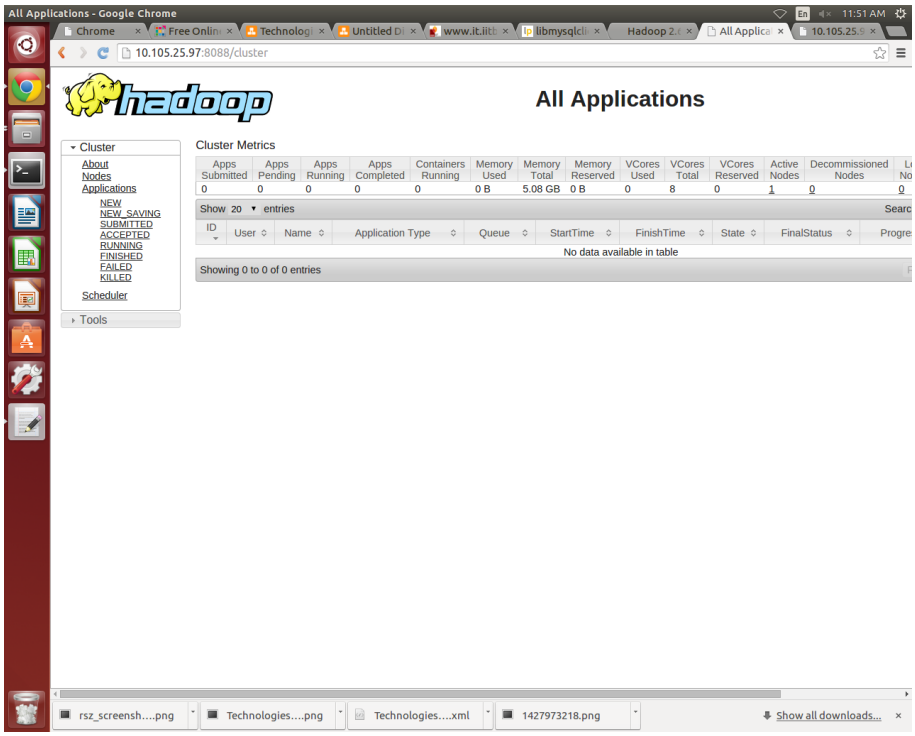


Figure 4.8: Hadoop MapReduce Cluster UI

4.2.9.4 Hadoop Map Reduce Node UI

Address : http://localhost:8042

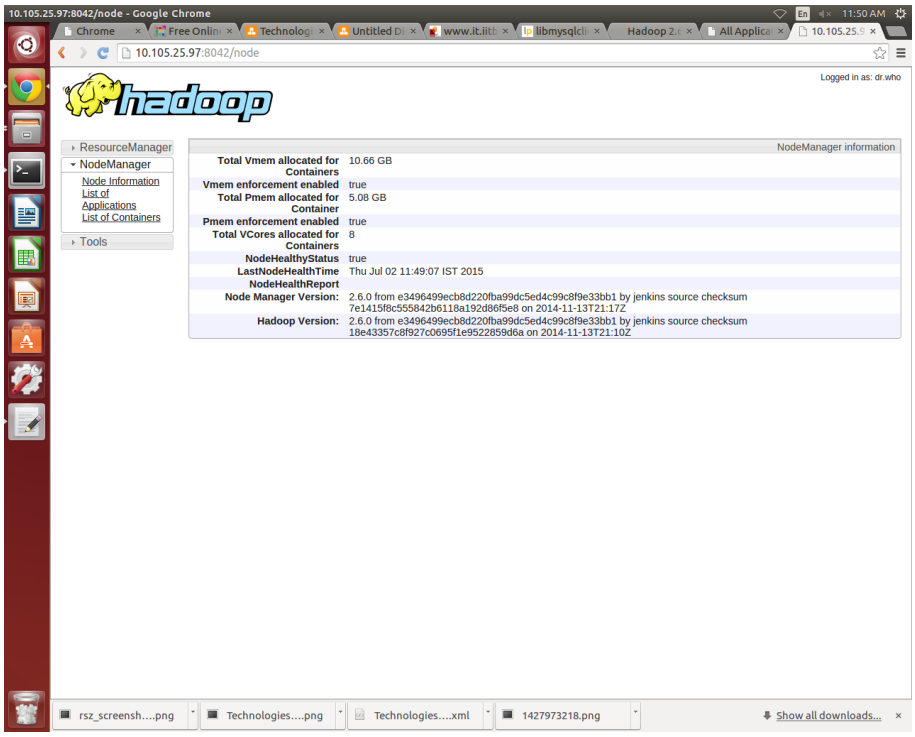


Figure 4.9: Hadoop MapReduce Cluster UI

4.2.10 Some known problems

1. Sometimes node becomes unhealthy, the only solution is to delete the hadoop_tmp/namenode and hadoop_tmp/datanode and reformat the NameNode.Be sure to stop the Hadoop before doing this.

2. Sometimes despite of starting the hadoop cluster properly, all the services may not start. In that case check the logs of the corresponding service in the $HADOOP_HOME/logs/ . According to the problem mentioned in the log try to solve the problem.

   Some known issues are :

   (a) The error may be of ports being previously bind, due to which the service may not be starting. The solution is to shutdown hadoop cluster , reboot the system and start the Hadoop cluster

   (b) Sometimes there may be a cluster mismatch between the namenode and the datanode . The solution is to delete the namenode and datanode directories and reformat the namenode as described above.

4.3 HIVE

Hive is used as a substitute to SQL for the Hadoop File System which makes it easy for people acquainted with SQL to query data from it instead of having to learn Map-Reduce.

4.3.1 Pre-requisites

Hadoop 2.x.0

4.3.2 Installation

1. Download the latest hive release from apache website.

   ```
   wget http://apache.mirrors.hoobly.com/hive/stable/
   apache-hive-0.13.0-bin.tar.gz
   ```

2. Extract the files and Set them up

   ```
   sudo tar -zxvf <hive install>
   sudo mv <hive-install> /usr/local/hive
   ```

3. Setup Environment Variables
   Add the following entries to .bashrc

   ```
   export HIVE PREFIX=/usr/local/hive
   export PATH=$PATH:$HIVE PREFIX/bin
   ```

4.3.3 Setting up MySQL as metastore instead of Derby

```
1.Install MySQL (if not installed previously)
$ sudo apt-get install mysql-server

2.Install the MySQL Java Connector
     $ sudo apt-get install libmysql-java

3.Create soft link for connector in Hive lib directory
$ ln -s /usr/share/java/mysql-connector-java.jar $HIVE_HOME/lib/mysql-connector-java.

4.Create the Initial database schema using the hive-schema-0.14.0.mysql.sql file ( or
$ mysql -u root -p
Enter password:
mysql> CREATE DATABASE metastore;
mysql> USE metastore;
mysql> SOURCE $HIVE_HOME/scripts/metastore/upgrade/mysql/hive-schema-0.14.0.mysql.sql

5.You also need a MySQL user account for Hive to use to access the metastore. It is v

mysql> CREATE USER 'hiveuser'@'%' IDENTIFIED BY 'hivepassword';
mysql> GRANT all on *.* to 'hiveuser'@localhost identified by 'hivepassword';
mysql>  flush privileges;



6. Create hive-site.xml ( If not already present) in $HIVE_HOME/conf folder with the
```

```
<configuration>
<property>
<name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:mysql://localhost/metastore?createDatabaseIfNotExist=true<;/value>
<description>metadata is stored in a MySQL server</description>
</property>
<property>
<name>javax.jdo.option.ConnectionDriverName</name>
<value>com.mysql.jdbc.Driver</value>
<description>MySQL JDBC driver class</description>
</property>
<property>
<name>javax.jdo.option.ConnectionUserName</name>
<value>hiveuser</value>
<description>user name for connecting to mysql server</description>
</property>
<property>
<name>javax.jdo.option.ConnectionPassword</name>
<value>hivepassword</value>
<description>password for connecting to mysql server</description>
</property>

## The value of the hive.metastore.uris depends on the system
or server on which the MySQL metastore is present, the port
remaining the same.

<property>
<name>hive.metastore.uris</name>
<value>thrif://localhost:9083</value>
<description>Thrift URI for the remote metastore used by the metastore client to conn

</property>

<property>
  <name>datanucleus.autoCreateSchema</name>
  <value>false</value>
</property>

<property>
  <name>datanucleus.fixedDatastore</name>
  <value>true</value>
</property>
</configuration>

7. We are all set now and we can start the hive console by
 $HIVE_HOME/bin/hive
```

4.3.4 Pre-requisite before launching Hive

In addition, you must create /tmp and /user/hive/warehouse (aka hive.metastore.warehouse.dir) and set them chmod g+w in HDFS before you can create a table in Hive. Commands to perform this setup:

```
$HADOOP_HOME/bin/hadoop fs -mkdir       /tmp
$HADOOP_HOME/bin/hadoop fs -mkdir       /user/hive/warehouse
$HADOOP_HOME/bin/hadoop fs -chmod g+w   /tmp
$HADOOP_HOME/bin/hadoop fs -chmod g+w   /user/hive/warehouse
```

You may find it useful, though it's not necessary, to set HIVE_HOME:

```
\$ export HIVE_HOME=<hive-install-dir>
```

Before launching the hive server we need to start the metastore thriftserver with the following command.

```
hive --service metastore &

## The command to check if metastore thrift server is running or not.

netstat -an | grep 9083
```

At the same time the MySQL server

4.3.5 Usage

1. Launch Hive

   ```
   $hive
   ```

2. CREATE

   ```
   CREATE TABLE books(id INT,name STRING,author STRING)
   ROW FORMAT DELIMITED FIELDS
   TERMINATED BY âĂŹ,âĂŹ STORED AS TEXTFILE;
   ```

3. LOAD

   ```
   LOAD DATA (LOCAL) INPATH "books.txt" INTO TABLE books;
   ```

4. SELECT

   ```
   SELECT * FROM books LIMIT 10;
   ```

4.4 Spark 1.2.1

- Apache Spark is an open-source data analytics cluster computing framework.
- Spark fits into the Hadoop open-source community, building on top of the Hadoop Distributed File System (HDFS).
- Spark is not tied to the two-stage MapReduce paradigm, and promises performance up to 100 times faster than Hadoop MapReduce, for certain applications.
- Follows the concept of a Resilient Distributed Dataset (RDD), which allows to transparently store data on memory and persist it to disc if itâĂŹs needed
- Provides Machine Learning Library MLLib for Data Analytics on the fly
- Spark provides interface to SQL and HQL (Hive Query Language) through SQLContext and HIVEContext libraries found in the pyspark.sql package.
- pyspark library is used for accessing spark through python with the help of sparkContext. For more information on sample code see spark programming guide.

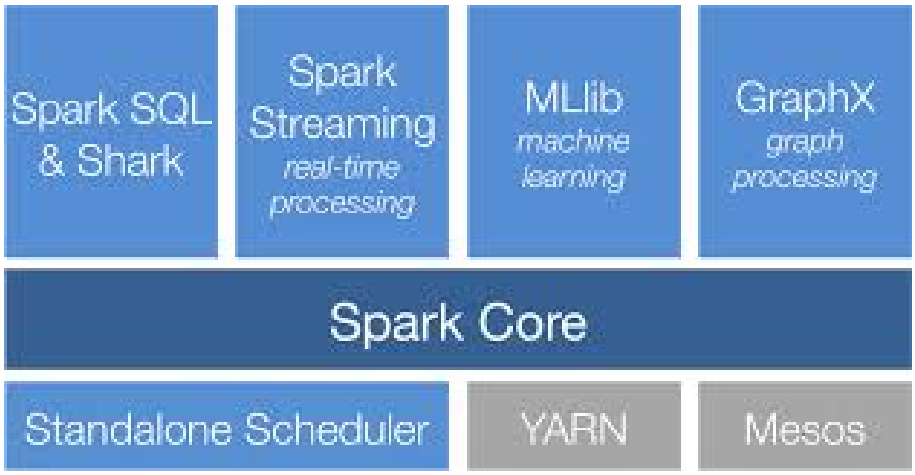


Figure 4.10: Name of the Figure

4.4.1 Steps for installation

4.4.1.1 Prerequisites Installation

1. Hadoop 2.6.0
2. Hive 1.2.0

4.4.1.2 Installation of commonly used python scipy tools

```
sudo apt-get -y install python-numpy python-scipy python-matplotlib ipython
ipython-notebook python-pandas python-sympy python-nose
```

4.4.1.3 Installation of scala

```
wget http://www.scala-lang.org/files/archive/scala-2.11.1.deb
sudo dpkg -i scala-2.11.1.deb
sudo apt-get -y update
sudo apt-get -y install scala
```

4.4.1.4 Installation of sbt

```
#One Line Command
wget http://scalasbt.artifactoryonline.com/scalasbt/sbt-native-packages/org
scala-sbt/sbt//0.12.3/sbt.deb

sudo dpkg -i sbt.deb
sudo apt-get -y update
sudo apt-get -y install sbt
```

4.4.1.5 Downloading and Unpacking spark

```
wget http://d3kbcqa49mib13.cloudfront.net/spark-1.2.1.tgz
tar -zxf spark-1.2.1.tgz
##Move the setup to /usr/local as
sudo mv /path to/spark-1.2.1.tgz /usr/local/spark
```

4.4.1.6 Building spark and Integrating with Hadoop and Hive

```
##One Line Command

./sbt/sbt -Phive -Phive-thriftserver -Pyarn -Phadoop-2.3
-Dhadoop.version=2.4.0 assembly
```

4.4.1.7 Clean-up

```
rm scala-2.11.1.deb
rm sbt.deb
rm spark-1.2.1.tgz
rm install.sh
```

4.4.2 Configuring User Files

1. Set the path variables in .bashrc as:

   ```
   export SPARK HOME=/usr/local/spark
   export PATH=$PATH:$SPARK HOME/bin
   export PATH=$PATH:$SPARK HOME/sbin
   ```

2. Add Configuration settings in /usr/local/spark/conf/ directory

```
cp spark-default.conf.template spark-default.conf
sudo vi spark-default.conf
##Add the following line to file
spark.driver.memory 5g
```

Copy the hive-site.xml from
$HIVE_HOME/conf/ directory to $SPARK_HOME/conf

3. In the $SPARK_HOME/bin/computepath.sh add the line before the last echo

   ```
   CLASSPATH="$CLASSPATH:/usr/share/java/mysql-connector-java-5.1.28.jar"
   ```

4. Run spark python shell:

   ```
   pyspark
   ```

4.5 MongoDB

4.5.1 Introduction

MongoDB is a open source, document-oriented, NoSQL database system.In the edx platform, the discussion form and some course related material is stored as collections of JSON-like documents in a MongoDB database. We have used local MongoDb database system to store the mongo dump files retrieved from the edX data package.

4.5.2 Installation Steps

1. Import the public key used by the package management system. sudo apt-key adv –keyserver hkp://keyserver.ubuntu.com:80 –recv 7F0CEB10

2. Create a list file for MongoDB.

   ```
   echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist
   10gen' | sudo tee /etc/apt/sources.list.d/mongodb.list
   ```

3. Reload local package database.
   sudo apt-get update

4. Install the MongoDB packages.
   sudo apt-get install mongodb-org

4.5.3 Basic Usage

After installing the MongoDB packages run the following commands to check whether the MongoDB is properly installed.

1. To start MongoDB : sudo start mongodb
2. To stop MongoDB : sudo stop mongodb
3. To run mongo shell: mongo
4. By default, mongo looks for a database server listening on port 27017 on the localhost interface.

4.6 MySQL 5.6

4.6.1 Introduction:

MySQL is a relational database management system (RDBMS), and ships with no GUI tools to administer MySQL databases or manage data contained within the databases.The data from the IITBX (eDX of IITB) i.e. the data from the IITBâĂŹs local server is the mysql dump files.We can process this data by imoprting it into HDFS using sqoop.

4.6.2 Installation Steps:

1. Go to the download page for the MySQL APT repository at

   `http://dev.mysql.com/downloads/repo/apt/.`

2. Select and download the release package for your platform
3. Install the downloaded release package with the following command, replacing platform-and-version-specific-package-name with the name of the downloaded package:

   ```
   sudo dpkg -i /PATH/platform-and-version-specific-package-name.deb
   ```

4. During the installation of the package, you will be asked to choose the versions of the MySQL server and other components (for example, the MySQL Workbench) that you want to install. If you are not sure which version to choose, do not change the default options selected for you. You can also choose none if you do not want a particular component to be installed. After making the choices for all components, choose Apply to finish the configuration and installation of the release package.
5. Use the following command to get the most up-to-date package information from the MySQL APT repository:

   ```
   shell> sudo apt-get update
   ```

6. `sudo apt-get install mysql-server`

7. Starting and Stopping the MySQL Server The MySQL server is started automatically after installation. You can check the status of the MySQL server with the following command:

   ```
   sudo service mysql status
   ```

   Stop the MySQL server with the following command:

   ```
   sudo service mysql stop
   ```

   To restart the MySQL server, use the following command:

   ```
   sudo service mysql start
   ```

4.6.3 Installing Additional MySQL Products and Components

1. You can use APT to install individual components of MySQL from the MySQL APT repository. Assuming you already have the MySQL APT repository on your system's repository list (see Adding the MySQL APT Repository for instructions), first, use the following command to get the latest package information from the MySQL APT repository:

   ```
   shell> sudo apt-get update
   ```

2. Install any packages of your choice with the following command, replacing package-name with name of the package (here is a list of available packages):

   ```
   sudo apt-get install package-name
   ```

   For example, to install the MySQL Workbench:

   ```
   sudo apt-get install mysql-workbench-community
   ```

   To install the shared client libraries:

   ```
   shell> sudo apt-get install libmysqlclient18
   ```

4.7 Django 1.6.5

4.7.1 Introduction:

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.It's free and open source.

4.7.2 Installation Steps:

1. sudo apt-get install python mysql-client mysql-server python-mysqldb
2. pip install django==1.6.5
   check whether django is installed properly
   python
   import django
   print(Django.get version())

4.8 Sqoop 1.4.5

4.8.1 Introduction

Sqoop is a tool designed to transfer data between Hadoop and relational databases. We used Sqoop to import data from a relational database management system (RDBMS) such as MySQL or Oracle into the Hadoop Distributed File System (HDFS). Sqoop automates most of the process, relying on the database to describe the schema for the data to be imported. Sqoop uses MapReduce to import and export the data, which provides parallel operation as well as fault tolerance. This section describes how to get started using Sqoop to move data between databases and Hadoop and provides reference information for the operation of the Sqoop commandline tool suite.

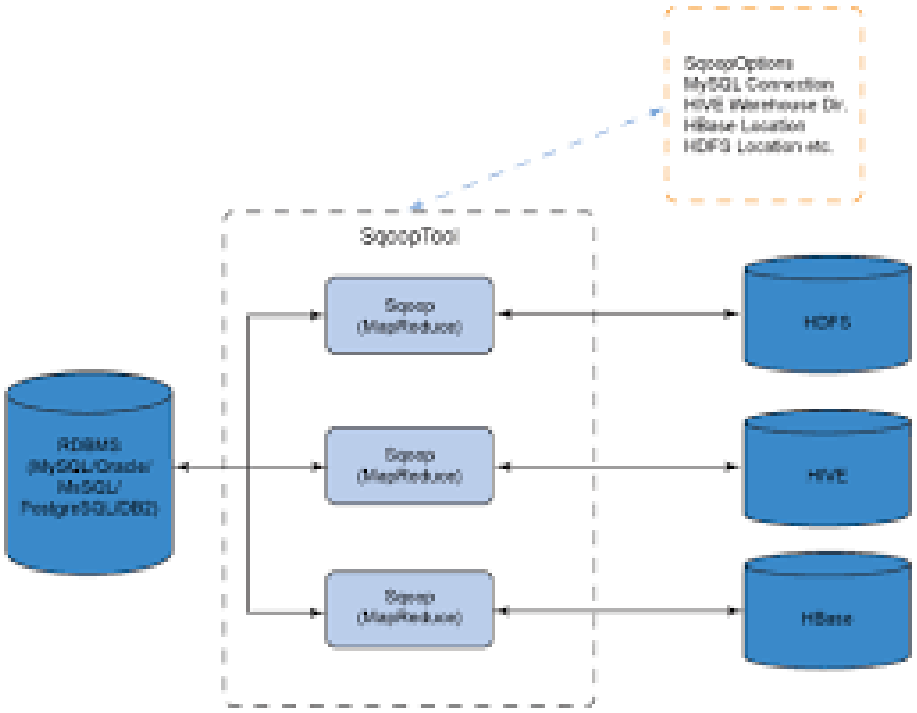


Figure 4.11: Name of the Figure

4.8.2 Stable release and Download

Sqoop is an open source software product of the Apache Software Foundation.Sqoop source code is held in the Apache GIT repository. You might clone the repository

using following command: git clone https://git-wip-us.apache.org/repos/asf/sqoop.git Latest stable release is 1.4.5. One can download it using following command: wget http://mirror.symnds.com/software/Apache/sqoop/1.4.5/sqoop-1.4.5.bin hadoop- 1.0.0.tar.gz . This command will download the latest stable release of sqoop 1.4.5 and store the tar file in the working directory.

4.8.3 Prerequisites

Before you can use Sqoop, a release of Hadoop must be installed and configured. Sqoop is currently supporting 4 major Hadoop releases - 0.20, 0.23, 1.0 and 2.0. We have installed Hadoop 2.6.0 and it is compatible with sqoop 1.4.5.We are using a Linux environment Ubuntu 14.04 to install and run sqoop. The basic familiarity with the purpose and operation of Hadoop is required to use this product.

4.8.4 Installation

To install the sqoop 1.4.5 we followed the given sequence of steps. 1. Unzip the tar file:

```
sudo tar -zxvf sqoop-1.4.5.bin hadoop1.0.0.tar.gz
```

2. Move sqoop-1.4.5.bin hadoop1.0.0 to sqoop:

```
sudo mv sqoop-1.4.5.bin hadoop1.0.0 /usr/local/sqoop
```

3. Set the SQOOP HOME path in bashrc file:

```
export SQOOP_HOME=/usr/lib/sqoop
export PATH=PATH:SQOOP_HOME/bin
```

4. Test your installation by typing:

```
sqoop help
```

4.9 Python Packages

This section describes the installation of different python packages required for various tasks such as query hive from client side, run bash commands, run Nosql(mongo) queries etc.

1. sasl : To install the pyhs2 package [24], sasl package has to be installed. It covers the authentication and security layers.
   To install it run :

   ```
   sudo pip install sasl

   ##If you get an error like this while installing sasl package :

   sasl/saslwrapper.cpp:21:23: fatal error: sasl/sasl.h: No such file or directory
   compilation terminated.

   error: command âĂŹgccâĂŹ failed with exit status 1
   ```

Then to rectify this error we installed the libsasl2-dev library using command :

```
sudo apt-get install libsasl2-dev
```

After installing this library, again try to install the sasl.

2. pyhs2 : This package enables us to run hive queries from python. Using pyhs2, we made connection with hiveserver2 and submit hive queries from client machine.
   To install pyhs2 run:

   ```
   sudo pip install pyhs2
   ```

3. pymongo : Using pymongo, we wrote mongo queries in python, made connection to mongoDB database system, run query and stored the result in a cursor.
   To install it run :

   ```
   sudo pip install pymongo
   ```

4. pygeoip : Using pygeoip, we found the location of the registered users from their IP addresses.Along with pygeoip we have used two databases GeoLiteCity.dat and GeoLiteCountry.dat. The locations found are used to visualize the number of users registered from a particular country on the world map.
   To install it run:

   ```
   sudo pip install pygeoip
   ```

5. pydoop.hdfs :Its a python package that allows to connect to an hdfs installation, rad and write files and get information on files, directories and global file properties. It acts as an hdfs api.
   To install it run:

   ```
   sudo pip install pydoop
   ```

6. mySQLdb: MySQLdb is an interface for connecting to a MySQL database server from Python. It implements the Python Database API v2.0 and is built on top of the MySQL C API.
   To install

   ```
   tar -xvf MySQL-python-1.2.2.tar
   cd MySQL-python-1.2.2
   python setup.py build
   python setup.py install
   ```

4.10 R Installation

R: (version 3.2.0)

```
sudo apt-get install r-base-core
```

R packages:

- **RMySQL:**

  ```
  install.packages("RMySQL")
  ```

- **googleVis:**

  ```
  install.packages("googleVis")
  ```

4.11 Multinode Cluster Configuration

4.11.1 Hadoop Multi Node Setup

Prerequisites

1. Installation and Configuration of Single node Hadoop :
   Install and Confiure Single node Hadoop which will be our Masternode.

2. Prepare your computer network (Decide no of nodes to set up cluster): Based on the helping parameters like Purpose of Hadoop Multinode cluster, Size of the dataset to be processed and Availability of Machines, you need to define no of Master nodes and no of Slave nodes to be configured for Hadoop Cluster setup.

3. Basic installation and configuration : Step 3A : Hostname identification of your nodes to be configured in the further steps. To Masternode, we will name it as HadoopMaster and to 2 different Slave nodes, we will name them as HadoopSlave1, HadoopSlave2 respectively in /etc/hosts directory. After deciding a hostname of all nodes, assign their names by updating hostnames (You can ignore this step if you do not want to setup names.) Add all host names to /etc/hosts directory in all Machines (Master and Slave nodes).

   ```
   # Edit the /etc/hosts file with following command
   sudo gedit /etc/hosts

   # Add following hostname and their ip in host table
   192.168.2.14    HadoopMaster
   192.168.2.15    HadoopSlave1
   192.168.2.16    HadoopSlave2
   ```

   Step 3B : Create hadoop as group and hduser as user in all Machines (if not created !!).

```
sudo addgroup hadoop
sudo adduser --ingroup hadoop hduser
```

If you require to add hdusers to sudoers, then fire following command

```
sudo usermod -a -G sudo hduser
```

OR
Add following line in /etc/sudoers/

```
hduser  ALL=(ALL:ALL) ALL
```

Step 3C : Install rsync for sharing hadoop source with rest all Machines,

```
sudo apt-get install rsync
```

Step 3D : To make above changes reflected, we need to reboot all of the Machines.

```
sudo reboot
```

Hadoop Configuration Steps

(a) Applying Common Hadoop Configuration :

However, we will be configuring Master-Slave architecture we need to apply the common changes in Hadoop config files (i.e. common for both type of Mater and Slave nodes) before we distribute these Hadoop files over the rest of the machines/nodes. Hence, these changes will be reflected over your single node Hadoop setup. And from the step 6, we will make changes specifically for Master and Slave nodes respectively.

Changes:

i. Update core-site.xml

Update this file by changing hostname from localhost to HadoopMaster

```
## To edit file, fire the below given command

hduser@HadoopMaster:/usr/local/hadoop/etc/hadoop$
sudo gedit core-site.xml

## Paste these lines into <configuration> tag OR
Just update it by replacing localhost with master
```

```
<property>
<name>fs.default.name</name>
<value>hdfs://HadoopMaster:9000</value>
</property>
```

ii. Update hdfs-site.xml

Update this file by updating repliction factor from 1 to 3.

```
## To edit file, fire the below given command

hduser@HadoopMaster:/usr/local/hadoop/etc/hadoop$
sudo gedit hdfs-site.xml

## Paste/Update these lines into <configuration> tag

<property>
<name>dfs.replication</name>
<value>3</value>
</property>
```

iii. Update yarn-site.xml

Update this file by updating the following three properties by updating hostname from localhost to HadoopMaster,

```
## To edit file, fire the below given command
hduser@HadoopMaster:/usr/local/hadoop/etc/hadoop$
sudo gedit yarn-site.xml

## Paste/Update these lines into <configuration> tag

<property>
<name>yarn.resourcemanager.resource-tracker.address</name>
<value>HadoopMaster:8025</value>
</property>
<property>
<name>yarn.resourcemanager.scheduler.address</name>
<value>HadoopMaster:8035</value>
</property>
<property>
<name>yarn.resourcemanager.address</name>
<value>HadoopMaster:8050</value>
</property>
```

iv. Update Mapred-site.xml

Update this file by updating and adding following properties,

```
## To edit file, fire the below given command
hduser@HadoopMaster:/usr/local/hadoop/etc/hadoop$
sudo gedit mapred-site.xml

## Paste/Update these lines into <configuration> tag
<property>
<name>mapreduce.job.tracker</name>
<value>HadoopMaster:5431</value>
</property>
<property>
<name>mapred.framework.name</name>
<value>yarn</value>
</property>
```

v. Update masters
Update the directory of master nodes of Hadoop cluster

```
## To edit file, fire the below given command

hduser@HadoopMaster:/usr/local/hadoop/etc/hadoop$
sudo gedit masters

## Add name of master nodes
HadoopMaster
```

vi. Update slaves
Update the directory of slave nodes of Hadoop cluster

```
## To edit file, fire the below given command

hduser@HadoopMaster:/usr/local/hadoop/etc/hadoop$
sudo gedit slaves

## Add name of slave nodes
HadoopSlave1
HadoopSlave2
```

(b) Copying/Sharing/Distributing Hadoop config files to rest all nodes âĂŞ master/slaves
Use rsync for distributing configured Hadoop source among rest of nodes via network.

```
# In HadoopSlave1 machine

sudo rsync -avxP /usr/local/hadoop/ hduser@HadoopSlave1:/usr/
local/hadoop/

# In HadoopSlave2 machine
sudo rsync -avxP /usr/local/hadoop/ hduser@HadoopSlave2:/usr/
```

```
local/hadoop/
```

The above command will share the files stored within hadoop folder to Slave nodes with location âĂŞ /usr/local/hadoop. So, you dont need to again download as well as setup the above configuration in rest of all nodes. You just need Java and rsync to be installed over all nodes. And this JAVA_HOME path need to be matched with

```
$HADOOP_HOME/etc/hadoop/hadoop-env.sh file of your Hadoop
distribution which we had already configured in Single node
Hadoop configuration.
```

(c) Applying Master node specific Hadoop configuration: (Only for master nodes) These are some configuration to be applied over Hadoop MasterNodes (Since we have only one master node it will be applied to only one master node.)

```
Step 6A : Remove existing Hadoop_data folder (which was created
while single node hadoop setup.

sudo rm -rf /usr/local/hadoop_tmp/

Step 6B : Make same (/usr/local/hadoop_tmp/hdfs) directory and
create NameNode ( /usr/local/hadoop_tmp/hdfs/namenode) directory

sudo mkdir -p /usr/local/hadoop_tmp/
sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode

Step 6C : Make hduser as owner of that directory.

sudo chown hduser:hadoop -R /usr/local/hadoop_tmp/
```

(d) Applying Slave node specific Hadoop configuration : (Only for slave nodes) Since we have three slave nodes, we will be applying the following changes over HadoopSlave1, HadoopSlave2 and HadoopSlave3 nodes.

```
Step 7A : Remove existing Hadoop_data folder
(which was created while single node hadoop setup)

sudo rm -rf /usr/local/hadoop_tmp/hdfs/


Step 7B : Creates same (/usr/local/hadoop_tmp/) directory/folder,
an inside this folder again Create DataNode
(/usr/local/hadoop_tmp/hdfs/namenode) directory/folder

sudo mkdir -p /usr/local/hadoop_tmp/
sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode


Step 7C : Make hduser as owner of that directory

sudo chown hduser:hadoop -R /usr/local/hadoop_tmp/
```

(e) Copying ssh key for Setting up passwordless ssh access from Master to Slave node :
To manage (start/stop) all nodes of Master-Slave architecture, hduser (hadoop user of Masternode) need to be login on all Slave as well as all Master nodes which can be possible through setting up passwrdless SSH login. (If you are not setting this then you need to provide password while starting and stoping daemons on Slave nodes from Master node).

```
Fire the following command for sharing public SSH key âĂŞ
$HOME/.ssh/id_rsa.pub file (of HadoopMaster node) to authorized_keys
 file of hduser@HadoopSlave1 and also on hduser@HadoopSlave1
 (in $HOME/.ssh/authorized_keys)

hduser@HadoopMaster: ~$
ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@HadoopSlave1

hduser@HadoopMaster: ~$
ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@HadoopSlave2
```

(f) Format Namenonde (Run on MasterNode) :

```
# Run this command from Masternode
hduser@HadoopMaster: usr/local/hadoop/$ hdfs namenode -format
```

(g) Starting up Hadoop cluster daemons : (Run on MasterNode)

```
Start HDFS daemons:
hduser@HadoopMaster:/usr/local/hadoop$ start-dfs.sh

Start MapReduce daemons:
hduser@HadoopMaster:/usr/local/hadoop$ start-yarn.sh
```

Instead both of these above command you can also use start-all.sh, but its now deprecated so its not recommended to be used for better Hadoop operations.

(h) Track/Monitor/Verify Hadoop cluster : (Run on any Node)
Verify Hadoop daemons on Master :

```
hduser@HadoopMaster: jps
```

Verify Hadoop daemons on all slave nodes :

```
hduser@HadoopSlave1: jps
hduser@HadoopSlave2: jps
```

If you wish to track Hadoop MapReduce as well as HDFS, you can also try exploring Hadoop web view of ResourceManager and NameNode which are usually used by hadoop administrators. Open your default browser and visit to the following links from any of the node.

For ResourceManager âĂŞ Http://HadoopMaster:8088
For NameNode âĂŞ Http://HadoopMaster:50070

4.11.2 Spark 1.2.1 multinode

1. sbin/start-master.sh - Starts a master instance on the machine the script is executed on.
2. sbin/start-slaves.sh - Starts a slave instance on each machine specified in the conf/slaves file.
3. sbin/start-all.sh - Starts both a master and a number of slaves as described above.
4. sbin/stop-master.sh - Stops the master that was started via the bin/startmaster.sh script.
5. sbin/stop-slaves.sh - Stops all slave instances on the machines specified in the conf/slaves file.
6. sbin/stop-all.sh - Stops both the master and the slaves as described above.

4.11.3 Some Important Commands

1. **ssh**: It is an important linux command to connect to other servers and perform actions remotely. If used with -X it enables X11 forwarding which helps in transferring the GUI from the server system to client.
   Usage:

   ```
   ssh [options] <username@ip of destination server>
   ```

2. **scp**: It is used to transfer data to and fro locally to server and vice versa.
   Usage:

   ```
   ssh [options] <location on host> <location on server>
   ```

3. Other Systems ports that are connected to that machines local host can be connected by using that machines IP and port address.

Chapter 5

OpenEdX Analytics System

5.1 Installation

Install and run the OpenEdX Analytics backend locally

1. Make a directory where you wish to install Analytics.Navigate to that directory
2. Clone the following repositories in that directory
   - edx/edx-analytics-data-api
   - edx/edx-analytics-pipeline
   - edx/edx-analytics-data-api-client
   - edx/edx-analytics-dashboard

```
cd {PATH_TO_DIRECTORY}
cd {PATH_TO_DIRECTORY}
git clone https://github.com/edx/edx-analytics-pipeline.git
git clone https://github.com/edx/edx-analytics-data-api.git
git clone https://github.com/edx/edx-analytics-data-api-client.git
git clone https://github.com/edx/edx-analytics-dashboard.git
```

3. Create virtual environments for running the repositories
   (a) Make a directory for storing virtual environments.Here we will make envs directory in home-directory

   ```
   cd
   mkdir envs
   ```

   (b) Make virtual environments

   ```
   cd ~/envs
   virtualenv edx-analytics-pipeline
   virtualenv edx-analytics-data-api
   virtualenv edx-analytics-data-api-client
   virtualenv edx-analytics-dashboard
   ```

4. Install MySQL locally and create database

```
sudo apt-get install mysql-server
mysql -u root -p

CREATE USER 'analytics'@'localhost' IDENTIFIED BY 'edx';
GRANT ALL PRIVILEGES ON *.* TO 'analytics'@'localhost';
FLUSH PRIVILEGES;
```

5.1.1 Install the Dependencies for each component

- Switch one by one to each environment.
- Install dependencies.
- Check them by running tests.

1. Install OpenEdX Analytics-pipeline

```
cd {PATH_TO_DIRECTORY}/analytics
cd edx-analytics-pipeline/
source ~/envs/edx-analytics-pipeline/bin/activate
make requirements
make test
deactivate
cd
```

2. Install OpenEdX analytics-data-api

```
source ~/envs/edx-analytics-data-api/bin/activate
cd {PATH_TO_DIRECTORY}/edx-analytics-data-api
make develop
./manage.py migrate --noinput
./manage.py migrate --noinput --database=analytics
./manage.py set_api_key edx edx
make validate
deactivate
cd
```

3. Install OpenEdX analytics-data-api-client

```
cd {PATH_TO_DIRECTORY}/edx-analytics-data-api-client/
source ~/envs/edx-analytics-data-api-client/bin/activate
pip install -r requirements.txt
make test
deactivate
cd
```

4. Install OpenEdX analytics-dashboard

```
cd {PATH_TO_DIRECTORY}/edx-analytics-dashboard/
source ~/envs/edx-analytics-dashboard/bin/activate
sudo apt-get update
sudo apt-get install gettext
sudo apt-get install npm
sudo apt-get install openjdk-7-jre
sudo apt-get install openjdk-7-jdk
sudo apt-get install libxml2-dev libxslt-dev python-dev zlib1g-dev
make develop
.\manage.py collectstatic
cd assets
mkdir dist
cp -R  js dist/
cd ..
make validate
deactivate
cd
```

5.1.2 Verify completion by running pipeline task locally

1. Create a credentials file for the pipeline

   ```
   cd {PATH_TO_DIRECTORY}
   vi mysql_creds
   ```

   Insert following lines in the file

   ```
   {
   "host": "127.0.0.1",
   "port": "3306",
   "username": "analytics",
   "password": "edx",
   "database": "analytics"
   }
   ```

2. Create configuration file override.cfg for pipeline

   ```
   cd edx-analytics-pipeline
   vi override.cfg
   ```

   Insert following lines in override.cfg

   ```
   [database-export]
   database = analytics
   credentials = {PATH_TO_DIRECTORY}/mysql_creds

   [database-import]
   database = edxprod
   ```

```
destination = s3://<bucket for intermediate hadoop products>/intermediate/databas
credentials = s3://<secrets bucket>/edxapp_prod_ro_mysql_creds

[event-logs]
expand_interval = 2 days
pattern = .*tracking.log-(?P<date>[0-9]+).*
source = s3://<bucket to where all tracking logs are synched>/tracking/

[hive]
warehouse_path = s3://<bucket for intermediate hadoop products>/warehouse/hive/

[manifest]
path = s3://<bucket for intermediate hadoop products>/user-activity-file-manifest
lib_jar = s3://<secrets bucket>/oddjob-1.0.1-standalone-modified.jar
input_format = oddjob.ManifestTextInputFormat

[enrollments]
blacklist_date = 2001-01-01
blacklist_path = /tmp/blacklist

[answer-distribution]
valid_response_types = customresponse,choiceresponse,optionresponse,multiplechoic
```

3. Acquire a log file (or create a dummy one)

```
cd
mkdir logs
cd logs
vi tracking.log
```

Copy data from
https://github.com/Sagarwal610/edx-analytic-pipeline-setup/blob/master/tracking.log

4. Run the API locally and query for results of the pipeline's aggregation

```
cd {PATH_TO_DIRECTORY}/edx-analytics-pipeline
source ~/envs/edx-analytics-pipeline/bin/activate
launch-task AnswerDistributionToMySQLTaskWorkflow \
 --local-scheduler --remote-log-level DEBUG --include  *tracking.log* \
 --src ~/logs --dest /tmp/answer_dist \
  --mapreduce-engine local --name test_task
deactivate
```

Now check the output in mysql database

```
mysql -u root -p

USE ANALYTICS;
SELECT COUNT(*) FROM answer_distribution;
```

Note :
-src defines source folder of log files

-dest defines output folder

If you get sql error that a particular column is not found then drop answer_distri table from analytics database.
Remove the file from /tmp/answer_distribution folder. Now again run the previous command.

Everytime you run the command an output file is generated .You have to delete that file before running the command again.

5. Run OpenEdX analytics data-api

```
cd {PATH_TO_DIRECTORY}/edx-analytics-data-api
source ~/envs/edx-analytics-data-api/bin/activate
./manage.py runserver --settings=analyticsdataserver.settings.local_mysql
```

Verify that API connects to the database

- Navigate to 127.0.0.1:8000 in your web browser
  - If the page does not display and you see ImproperlyConfigured: Error loading MySQLdb module in the logs, run: 'pip install mysql-python
  - If the page indicates a 401 access forbidden error, you need to rerun:
    ./manage.py set_api_key edx edx
- Get result
  - Click on api
  - Click on /api/v0/problems/problem_id/answer_distribution/
  - Enter problem id
    In this enter module_id from your logs
    if you are using dummy log file then enter -
    i4x://edX/DemoX-S/problem/a58470ee54cc49ecb2bb7c1b1c0ab43a

Note:

- **If you are behind a proxy :**
  - **add following lines in activate file of each virtual environment. This file is located in bin of virtualenvironment**

```
export HTTP_PROXY="http://username:password@yourdomain.com:PORT"

export HTTPS_PROXY="https://username:password@yourdomain.com:PORT"
```

You can also add these lines in .bashrc file in your home directory.

And use sudo -E instead of sudo .

- **Configure git to work behind proxy**

```
git config --global http.proxy http://username:password@yourdomain.com:PO

git config --global https.proxy https://username:password@yourdomain.com:
```

- **Error in installing npm :**

  Try installing node.js and nodejs-legacy

```
sudo apt-get install nodejs
sudo apt-get install nodejs-legacy
```

- **OpenEdX Analytics Pipeline configuration**

  Configuration for pipeline can be done by modifying override.cfg file

  - **Configure [event-logs]**

    Change source to
    `hdfs://IP:PORT/path_to_logFilesDirectory`

    `for reading logs from hdfs file system.`

- **Authentication problem in data-api**

  If you get authentication error in data-api

  - **Check local_mysql.py file in analyticsdataserver/settings folder of edx-analytics-data-api and configure databases**
  - **Activate edx-analytics-data-api environment ans run ./manage.py syncdb in edx-analytics-data-api folder**

- **Logging in OpenEdX analytics-dashboard**

```
cd
source envs/edx-analytics-dashboard/bin/activate
cd {PATH_TO_DIRECTORY}/edx-analytics-dashboard
./manage.py runserver 9000
```

5.2 Frontend

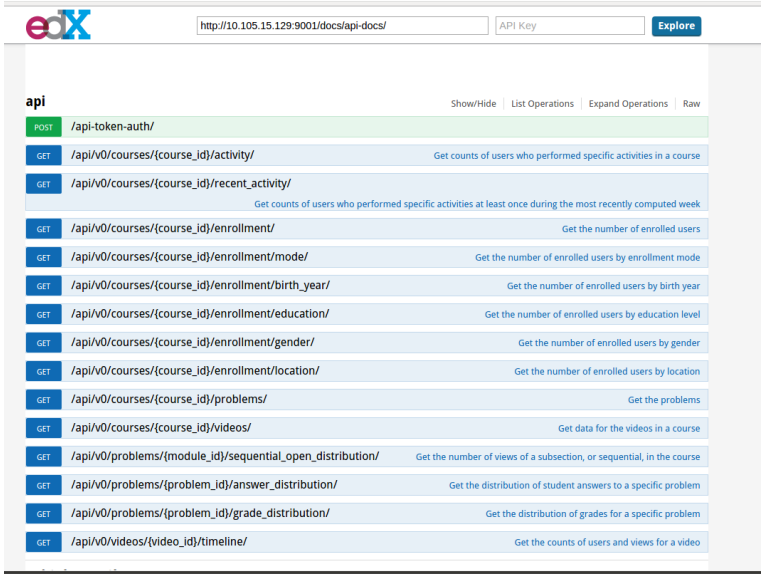


Figure 5.1:

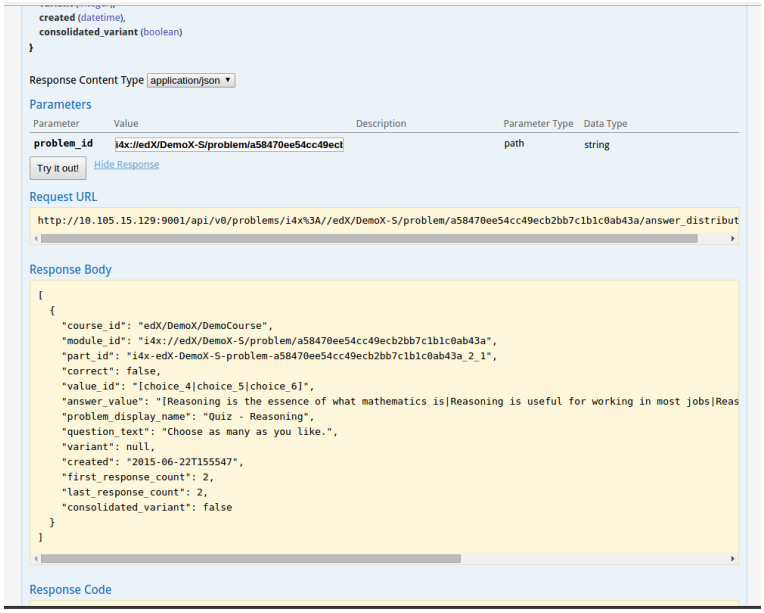


Figure 5.2:

Chapter 6

Our Pipeline

6.1 Architecture

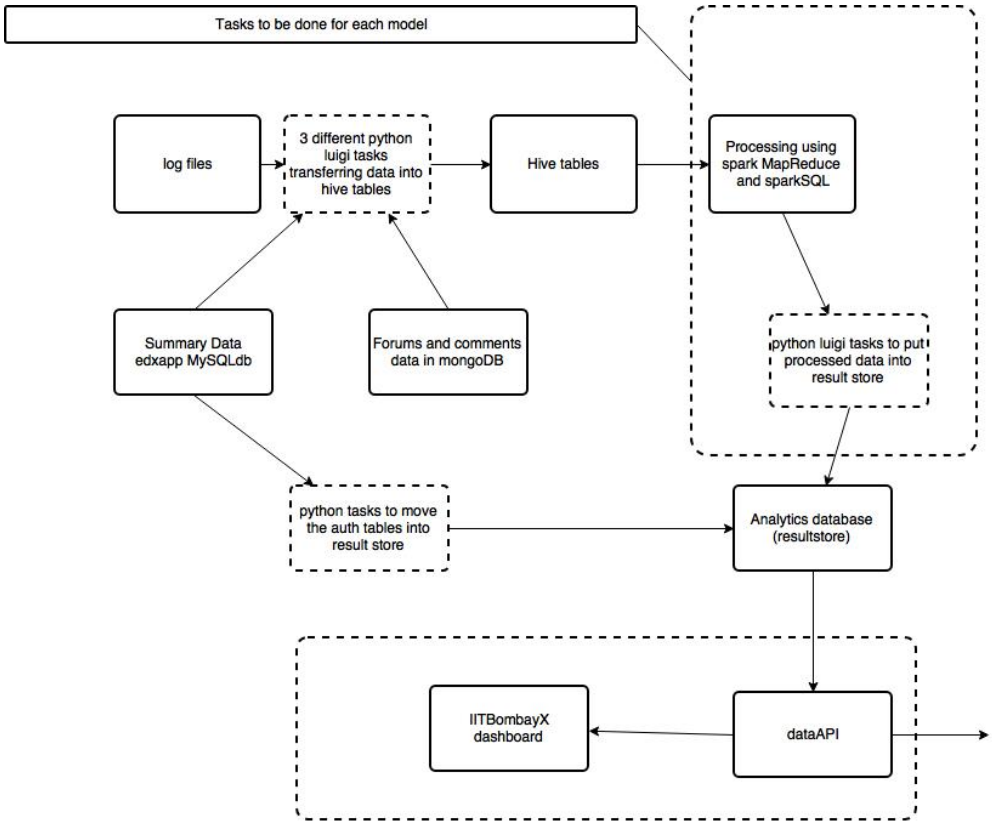


Figure 6.1: Architecture

6.1.1 How the system works

The data is extracted from the log files and the summary data, it is cleaned and stored in the SQL database which is then transferred into hive database. The other appropriate data from the summary database is directly transferred to the result store (Analytics Database). The data in the hive tables is processed using Spark MapReduce and stored into the result store depending upon the model being processed.
This result store is then used by the dashboard using the data API to visualise the results.

6.2 Components of the Pipeline

In this section we discuss about the the different stages of the pipeline one by one in detail.

6.2.1 The ETL stage

6.2.1.1 In the actual edX analytics pipeline

In the actual pipeline whenever a batch process is run for populating a particular model , the entire log data is scanned and cleaned for each model searching for the appropriate events and data. Then the data is processed using Hadoop MapReduce to populate the ResultStore.
This data is stored in Amazon S3 Cloud and hence an EMR CLuster has to be set up on Amazon S3 to process the required data.

6.2.1.2 In Our Pipeline

In our pipeline we have used Python Luigi Tasks to clean the data and store it for only once new data enters the system. Every time a batch process is run the required data is directly fetched from the Hive Database and thus the repitiitve cleaning is not required saving processing time.
Due to this there is no need to store data on Amazon S3 Cloud , and the local servers can be used to store and retrive the data as and when required.

6.2.2 The Python Code

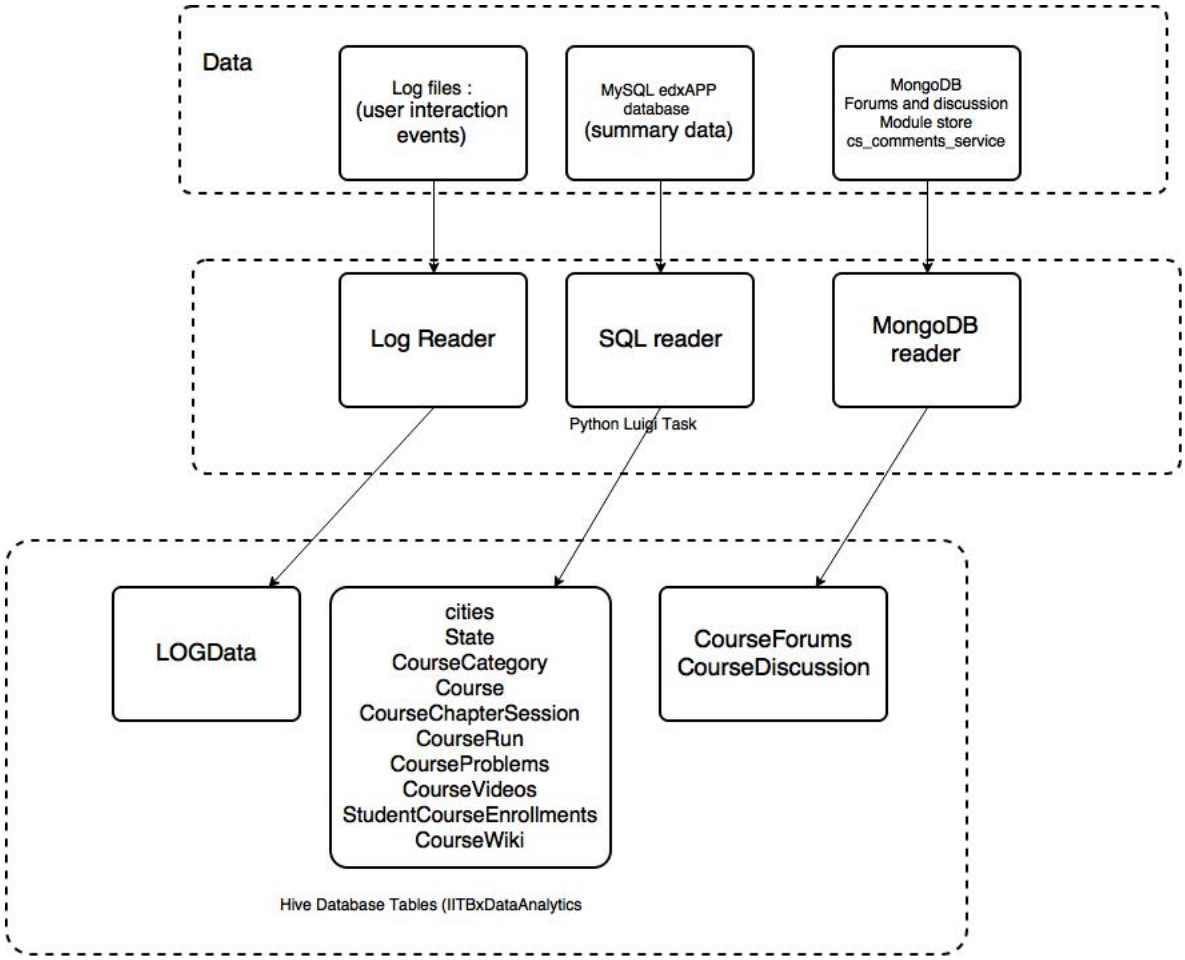


Figure 6.2: Cleaning task

6.2.2.1 Data Source

The data is taken from 3 diffrent sources:

1. ***The Log Data*** **: This is the data that gets stored on milliseconds basis as JSON Records based on the User Interaction with the system.**
2. ***The SQL Data*** **: This is the summary data that stores the state of the LMS in tables. The name of the Database is edxapp. It stores information ranging from User Authorisation to Student Problem Details. The Detailed Schema is as follows:**

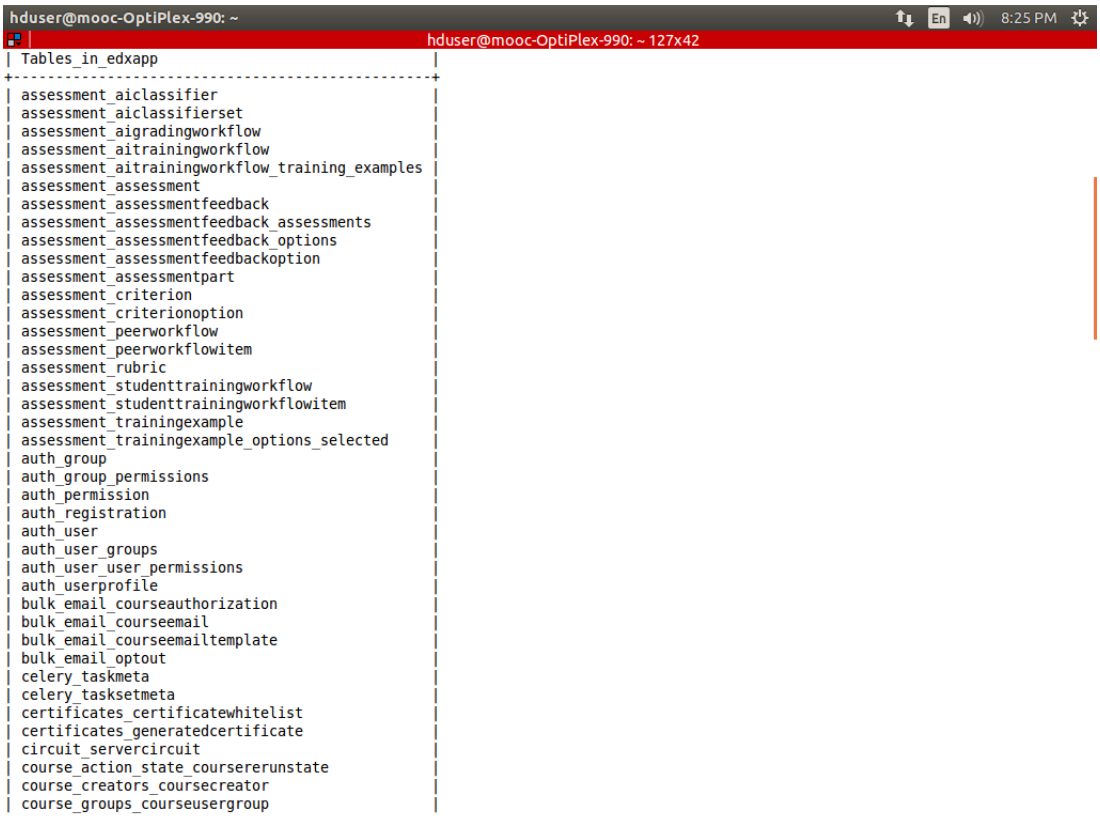


Figure 6.3: Edxapp Tables Part 1

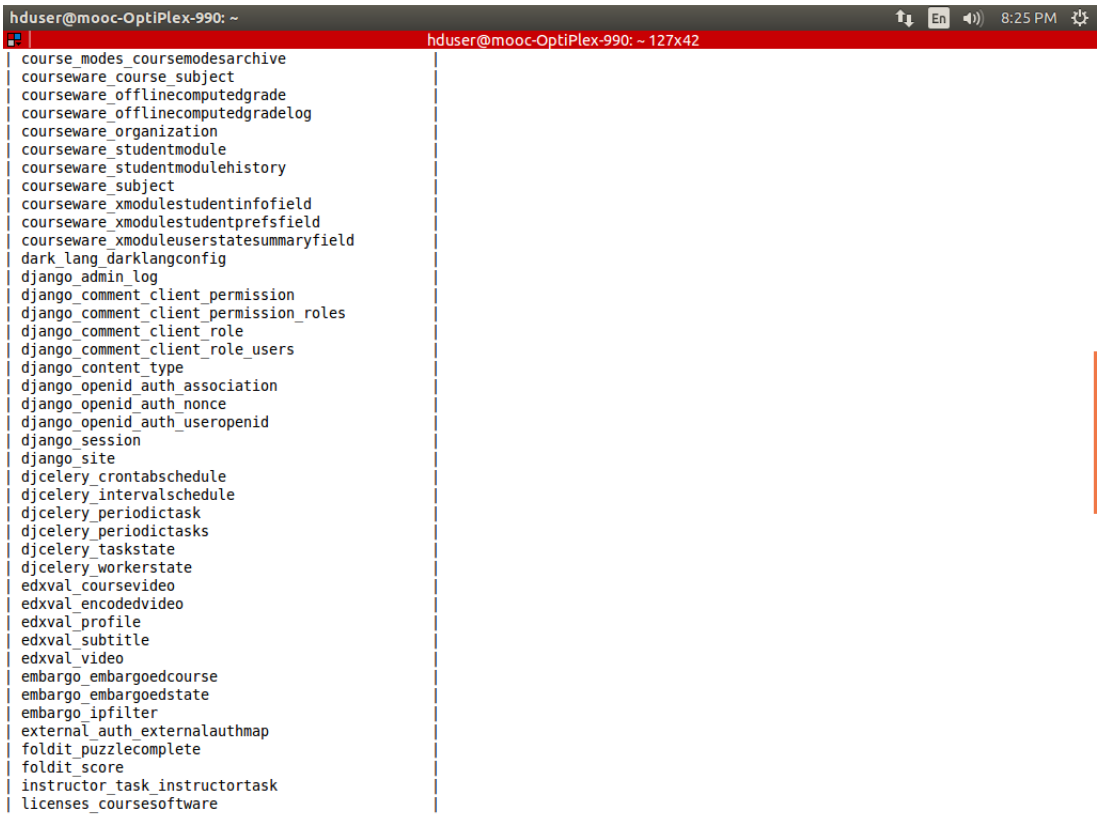


Figure 6.4: Edxapp Tables Part 2

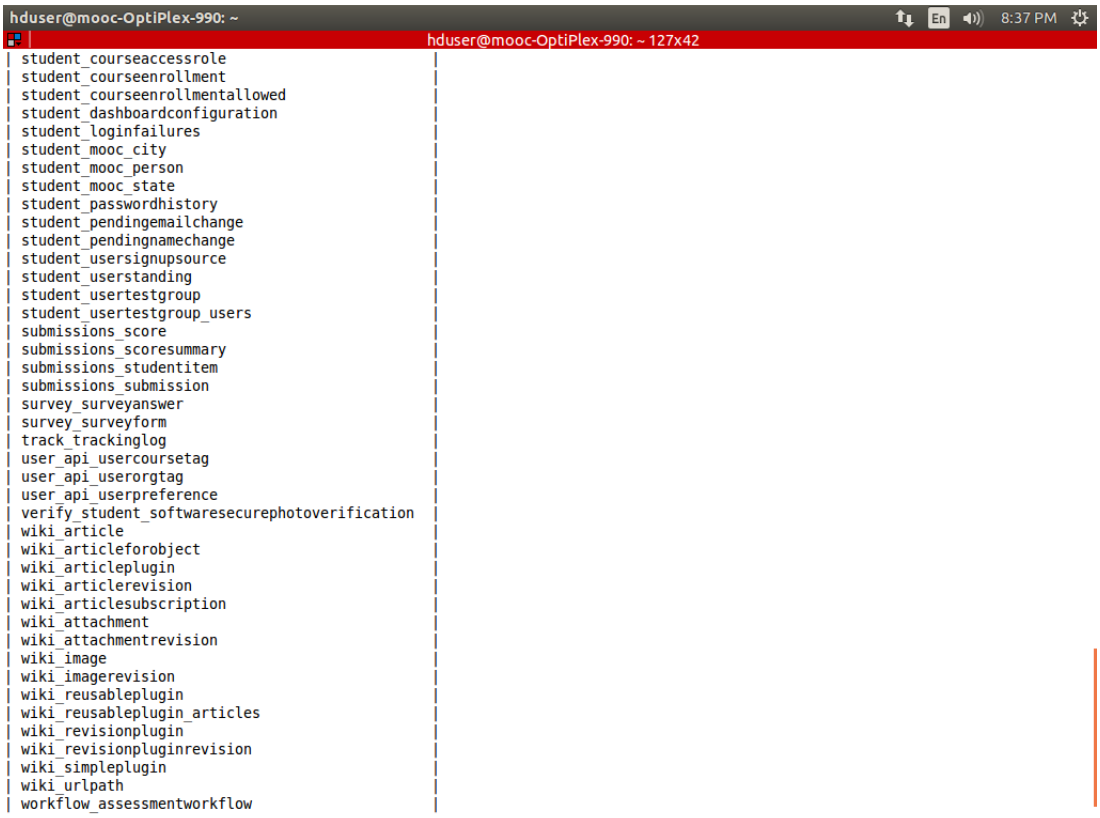


Figure 6.5: Edxapp Tables Part 3

3. ***The MongoDB Data*** **: This is the Unstructured Data stored as JSON Collections. It consists data about CourseDiscussions, CourseForums, CourseGrading Policy etc.**

6.2.2.2 The Processing

The Python Project Consists of Diffrent Python Codes which can be run as Combined Luigi Process or Standalone Luigi Process. The Processing comprises of removing the unwanted data and organiznig the data in a meaningful manner.
All this Data is first inseretd into MySQL Database and the same schema is used to transfer the data to HIVE Tables using SQOOP.

- ***The LogReader*** **: This Python Code reads the JSON format Log Files and based on the events it classifies them. The Table the data is stored in has 58 fields. It is not necessary that all the fields are filled pertaining to a particular event record.**
  The Schema of The Log Table is :

Field	Type	Null	Key	Default	Extra
sessionId	bigint(20)	NO	PRI	NULL	auto_increment
sessionSysName	varchar(50)	YES		NULL	
lmsName	varchar(6)	NO		NULL	
orgName	varchar(50)	YES		NULL	
courseName	varchar(50)	YES		NULL	
courseRun	varchar(150)	YES		NULL	
lmsUserId	bigint(20)	YES		NULL	
userName	varchar(50)	YES		NULL	
agent	varchar(255)	YES		NULL	
hostName	varchar(50)	YES		NULL	
ipAddress	varchar(50)	YES		NULL	
createDateTime	timestamp	YES		NULL	
eventType	varchar(100)	YES		NULL	
eventSource	varchar(50)	NO		NULL	
eventName	varchar(255)	YES		NULL	
dataSource	varchar(5)	YES		NULL	
oldVideoSpeed	float	YES		NULL	
currVideoSpeed	float	YES		NULL	
oldVideoTime	float	YES		NULL	
currVideoTime	float	YES		NULL	
videoNavigType	varchar(15)	YES		NULL	
oldGrade	float	YES		NULL	
currGrade	float	YES		NULL	
maxGrade	float	YES		NULL	
attempts	int(11)	YES		NULL	
maxNoAttempts	int(11)	YES		NULL	
hintAvailable	varchar(10)	YES		NULL	
hintUsed	longtext	YES		NULL	
currPosition	int(11)	YES		NULL	
oldPosition	int(11)	YES		NULL	
chapterSysName	varchar(45)	YES		NULL	
chapterTitle	varchar(255)	YES		NULL	
sessSysName	varchar(45)	YES		NULL	
sessTitle	varchar(255)	YES		NULL	
moduleSysName	varchar(45)	YES		NULL	
moduleTitle	varchar(255)	YES		NULL	
answerChoice	longtext	YES		NULL	
success	varchar(10)	YES		NULL	
done	varchar(6)	YES		NULL	
enrolmentMode	varchar(8)	YES		NULL	
totDurationInSecs	int(20)	YES		NULL	
eventNo	smallint(6)	YES		NULL	

Figure 6.6: Schema of Log Data

An Example Conversion is:

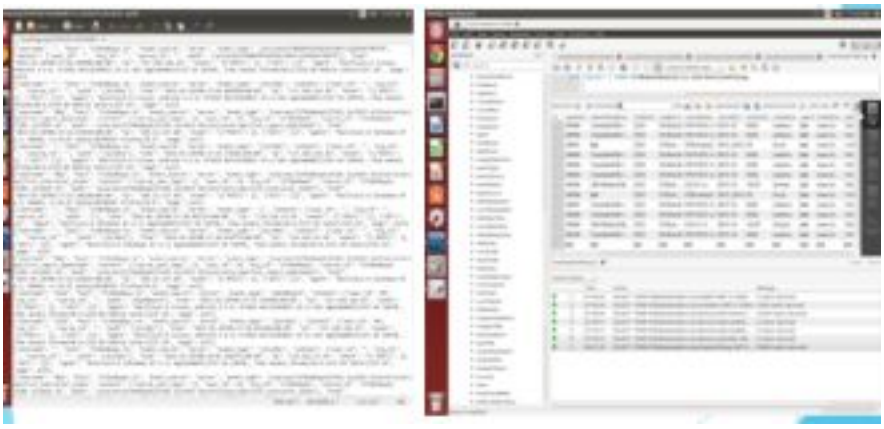


Figure 6.7: Json Data to Organised MySQL Data

The Classififed Events Are :

```
==========================================
 * load_video -  11
```

```
* pause_video -  12
* play_video 13
* seek_video 14
* speed_change_video 15
* stop_video 16
* hide_transcript 17
* show_transcript 18
* save_user_state 19
* transcript_translation 20
* transcript_download 21
* /transcript/translation/ 22 (in edx properties save_video_position)
*
* /transcript/download") 23
*
* PROBLEM 45 - 65
* ================================================
* problem_check 45
* problem_check_fail 46
* problem_reset 47
* problem_rescore 48
* problem_rescore_fail 49
* problem_save 50
* problem_show 51
* reset_problem 52
* reset_problem_fail 53
* save_problem_success 54
* problem_graded 55
* showanswer 56
* save_problem_fail 57
* problem_get 58
*
* NAVIGATION  80 - 105
* ================================================
* seq_goto 80
* seq_next 81
* seq_prev 82
* page_close 83
* goto_position 84
*  /dashboard 85
*  / 86
* /jsi18n/ 87
* /i18n.js 88
* jump_to_discussion 89
* progress 90
* view_courses 91
* logout  92
* how_it_works 93
* calculate 94
```

```
* city 95
* jump_to_vertical 96
* login 97
*
* DISCUSSION 120 - 135
* ================================================
* threads 120
* users 121
* reply  122
* thread_create 123
* upvote  124
* flagAbuse 125
* follow 126
* unfollow 127
* upload 128
* forum_searched 129
*
* COURSE 150 - 160
* ================================================
* courseware 150
* chapter 151
* session 152
*
* courses_chapter 153
* courses_access 154
*
*
* WIKI   175 - 180
* ================================================
* wiki  175
*
*
*INSTRUCTOR EVENTS 195 - 205
*================================================
*
*list_instructor_tasks 195
*list-staff 196
*dump-graded-assignments-config 197
*
*
*OPEN ASSESMENT                        220 - 240
* ========================================================
* render_self_assessment 220
* render_submission 221
* render_leaderboard 222
* render_peer_assessment 223
* render_message 224
* render_grade 225
```

```
* render_student_training 226
* save_submission 227
* submit_feedback 228
* training_assess 229
* peer_assess 230
* submit 231
* render_staff_info 232
*
*
* ENROLLMENT  255 - 265
* ================================================
*
*  edx.course.enrollment.activated     255
*  edx.course.enrollment.deactivated   256
*  edx.course.enrollment.mode_changed 257
*  change_enrollment 258
*  /register 259
*  /create_account 260
*
*  TEXTBOOK INTERACTION EVENTS 280 - 285
*  ============================================
*  book  280
*
*
*
* ADMIN 290 - 300
* ============================================
* /admin 290
* pref_status 291
* password_reset 292
* password_reset_confirm 293
*
*/
```

- ***The SQLDataReader*** **: This Python Code reads diffrent summary tables and produces the following tables with the table name suggesting the information stored:**

otherTitle	varchar(50)	YES		NULL	
otherType	varchar(50)	YES		NULL	
anonymous	varchar(1)	YES		NULL	
anonymousToPeers	varchar(1)	YES		NULL	
eduLevel	varchar(30)	YES		NULL	
gender	varchar(1)	YES		NULL	
commentableId	varchar(50)	YES		NULL	
commentType	varchar(50)	YES		NULL	
commentSysId	varchar(50)	YES		NULL	
aadhar	varchar(45)	YES		NULL	
problemSubmissionTime	timestamp	YES		NULL	
hintMode	varchar(50)	YES		NULL	
currentSeekTime	float	YES		NULL	
queryText	longtext	YES		NULL	
noOfResults	smallint(6)	YES		NULL	
lastModDateTime	timestamp	YES		NULL	

Figure 6.8: Summary Data

Field	Type	Null	Key	Default	Extra
otherId	int(11)	NO	PRI	NULL	auto_increment
lmsName	varchar(6)	YES		NULL	
orgName	varchar(50)	YES		NULL	
courseName	varchar(50)	YES		NULL	
title	varchar(255)	YES		NULL	
htmlSysName	varchar(50)	YES		NULL	
type	varchar(45)	YES		NULL	
verticalSysName	varchar(50)	YES		NULL	
chapterSysName	varchar(50)	YES		NULL	
sessionSysName	varchar(50)	YES		NULL	
fileName	varchar(45)	YES		NULL	

Figure 6.9: Other Details about courses

Field	Type	Null	Key	Default	Extra
problemId	bigint(20)	NO	PRI	NULL	auto_increment
lmsName	varchar(6)	YES		NULL	
orgName	varchar(50)	YES		NULL	
courseName	varchar(50)	YES		NULL	
chapterSysName	varchar(50)	YES		NULL	
sessionSysName	varchar(50)	YES		NULL	
quizSysName	varchar(50)	YES		NULL	
quizTitle	longtext	YES		NULL	
quizType	varchar(50)	YES		NULL	
quizWeight	float(6,2)	YES		NULL	
noOfAttemptsAllowed	int(3)	YES		NULL	
quizMaxMarks	float(6,2)	YES		NULL	
hintAvailable	smallint(6)	YES		NULL	
correctChoice	smallint(6)	YES		NULL	
hintMode	varchar(15)	YES		NULL	

Figure 6.10: Data related to course quizzes and problems

Field	Type	Null	Key	Default	Extra
courseRunid	int(11)	NO	PRI	NULL	auto_increment
lmsName	varchar(6)	NO		NULL	
orgName	varchar(50)	YES		NULL	
courseName	int(50)	NO		NULL	
courseRun	varchar(150)	YES		NULL	
willbeGraded	varchar(1)	YES		NULL	
gradePass	float(5,2)	YES		NULL	
actualPrice	int(11)	YES		NULL	
currencyCode	varchar(3)	NO		NULL	
startDate	date	YES		NULL	
endDate	date	YES		NULL	

Figure 6.11: Currently Running Course Details

Field	Type	Null	Key	Default	Extra
videoId	int(11)	NO	PRI	NULL	auto_increment
lmsName	varchar(6)	YES		NULL	
orgName	varchar(50)	YES		NULL	
courseName	varchar(50)	YES		NULL	
chapterSysName	varchar(50)	YES		NULL	
videoSysName	varchar(255)	YES		NULL	
videoUTubeId	varchar(50)	YES		NULL	
videoDownload	tinyint(1)	YES		NULL	
videoTrackDownLoad	tinyint(1)	YES		NULL	
videoTitle	varchar(255)	YES		NULL	
videoUTubeId075	varchar(50)	YES		NULL	
videoUTubeId125	varchar(50)	YES		NULL	
videoUTubeId15	varchar(50)	YES		NULL	

Figure 6.12: Data About Videos Related to Courses

Field	Type	Null	Key	Default	Extra
wikiId	int(11)	NO	PRI	NULL	auto_increment
lmsName	varchar(6)	YES		NULL	
orgName	varchar(50)	YES		NULL	
courseName	varchar(50)	NO		NULL	
wikiSlug	varchar(50)	NO		NULL	
lmsWikiId	bigint(20)	YES		NULL	
createdDate	datetime	YES		NULL	
lastModDate	datetime	YES		NULL	
lastRevId	int(11)	YES		NULL	
ownerId	bigint(20)	YES		NULL	
groupId	bigint(20)	YES		NULL	
groupRead	tinyint(4)	YES		NULL	
groupWrite	tinyint(4)	YES		NULL	
otherRead	tinyint(4)	YES		NULL	
otherWrite	tinyint(4)	YES		NULL	

Figure 6.13: Data about Wiki Related To Courses

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
lmsUserId	int(11)	NO	MUL	NULL	
orgName	varchar(64)	NO		NULL	
courseName	varchar(255)	NO		NULL	
courseRun	varchar(45)	YES		NULL	
role	varchar(64)	NO		NULL	

Figure 6.14: Data about Access Roles Pertaining to Courses

Field	Type	Null	Key	Default	Extra
enrolId	bigint(20)	NO	PRI	NULL	auto_increment
lmsName	varchar(6)	YES		NULL	
orgName	varchar(50)	YES		NULL	
courseName	varchar(45)	YES		NULL	
courseRun	varchar(50)	YES		NULL	
lmsUserId	bigint(20)	YES		NULL	
enrolmentDate	timestamp	YES		NULL	
active	varchar(1)	YES		NULL	
mode	varchar(15)	YES		NULL	

Figure 6.15: Data about Course Enrollment

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	NULL	auto_increment
lmsName	varchar(50)	YES		NULL	
orgName	varchar(50)	YES		NULL	
courseName	varchar(50)	YES		NULL	
courseRun	varchar(255)	YES		NULL	
lmsUserId	bigint(20)	YES		NULL	
lmsUserName	varchar(100)	YES		NULL	
moduleType	varchar(45)	YES		NULL	
moduleSysName	varchar(255)	YES		NULL	
score	int(3)	YES		NULL	
maxScore	int(3)	YES		NULL	
noOfAttempts	int(2)	YES		NULL	
hintUsed	varchar(255)	YES		NULL	
hintAvailable	varchar(255)	YES		NULL	
state	longtext	YES		NULL	
goals	varchar(50)	YES		NULL	
createDateTime	timestamp	YES		NULL	
lastModDateTime	timestamp	YES		NULL	
totSessDuraInSecs	int(11)	YES		NULL	
done	varchar(45)	YES		NULL	

Figure 6.16: Data About Course Grades

Field	Type	Null	Key	Default	Extra
userId	bigint(10)	NO	PRI	NULL	auto_increment
lmsUserId	bigint(20)	NO	UNI	NULL	
lmsName	varchar(6)	NO		NULL	
orgName	varchar(50)	YES		NULL	
name	varchar(100)	NO		NULL	
gender	varchar(1)	YES		NULL	
registrationDate	date	YES		NULL	
emailId	varchar(100)	NO		NULL	
mothertounge	varchar(2)	YES		NULL	
highestEduDegree	varchar(6)	YES		NULL	
goals	longtext	YES		NULL	
city	varchar(120)	YES		NULL	
state	varchar(120)	YES		NULL	
active	smallint(6)	NO		0	
firstAccesDate	datetime	YES		NULL	
lastAccessDate	datetime	YES		NULL	
allowCert	smallint(6)	YES		NULL	
yearOfBirth	smallint(6)	YES		NULL	
pincode	int(11)	YES		NULL	
aadharId	varchar(45)	YES		NULL	

Figure 6.17: Data about Users

- *The MongoDBReader*: It reads the data from the Mongo Database and genrates the CourseForum and CourseDiscussions Table.

6.2.3 The Analysis Models

In our system we attempt to incorporate the existinfg models on OpenEdx-Insights and add some of our own indigenious Data Models some of which are based on Statistical Data Mining, while some on Machine Learning

The OpenEdxInsight has the following 3 Analysis Models:

1. Course Enrollment : The Enrollment statistics of a particular course based on age, gender, location etc.
2. Course Engagement : The Engagement of Users in Diffrent Course Resources (Quiz, Video, Forums etc.) over time.
3. Course Performance : Statistical Analysis of Performance of students in a particualr course over time;

6.2.3.1 The Process

Data is extracted from the Hive Tables whose source is Log Data and summary Data depending upon the model.
The Data is then processed using Spark and SparkSQL to process the data in a form which can be inserted into the Analytics Data Store for visualisation.

6.2.4 The Course Enrollment Model in Detail

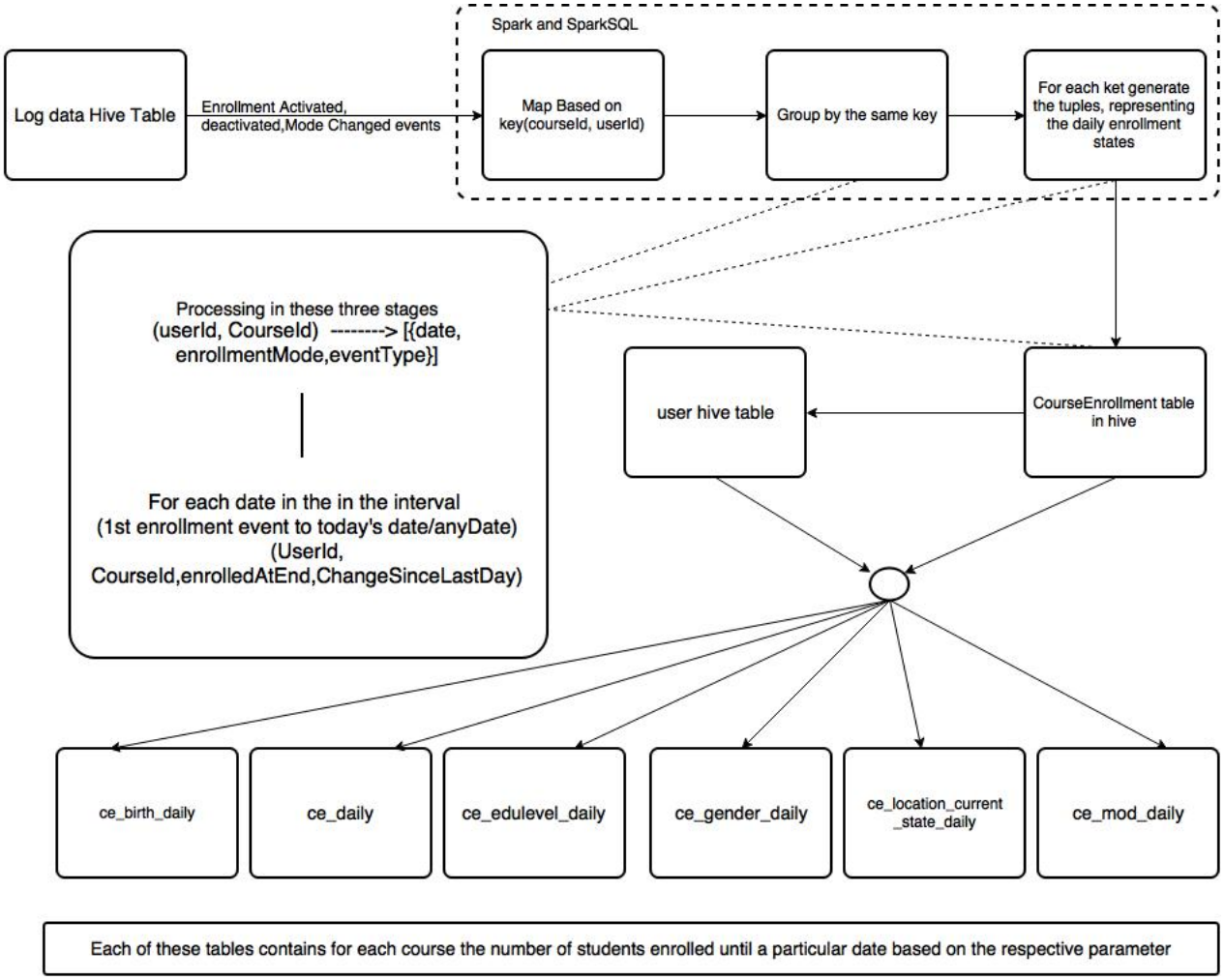


Figure 6.18: Detail Course Enrollment Model

6.2.5 Our ResultStore : Analytics

Structure is the same as in the OpenEdx Insight. The different tables are:

6.3 Log Processing

6.3.1 The Need

The Need for Log Processing in the present scenario is , because it records the momentry footsteps of the users and shows whatever the user does. Thus helping to map the Students Learning Curve.

6.3.2 The Significance

Diffrent Events have diffrent fields showing diffrent information ranging from the time the user spent on a particular video, number of questions he answered correct, the number of chapters he acessed etc. helping to genrate the diffrent models.
Each Log Record contains information pertaining to a user and a resource (Chapter, Quiz, Video, Problem, Book etc.) thus giving the approriate information which is recorded in th Tables.

6.4 Challenges Faced

The Challenge is that we don't have acess to the enitre log Record Base.

6.5 Advantages

The Advantage over OpenEdx Insight System:

1. **We are doing the cleaning process per batch process only once for all models compared to repitive scanning and cleaning in Edx Analytics.**
2. **There is no requirement of Amazon S3 Cloud for storage of data thus economical.**
3. **Usage of Spark instead of Hadoop for Processing MapReduce thus saving time.**

Chapter 7

Our Dashboard

This section contains information about the data that IITBombayX Data Analytics presents. A detailed report about the graphs, metrics and reports that can be accessed in IITBombayX Data Analytics has been provided along with information about the computation of the values that are reported.

7.1 Enrollment Activity

How many students are enrolled in the course? IITBombayX Data Analytics helps you monitor how many students are enrolled for the course and how that number changes over time.

7.1.1 Gaining Insight into Course Enrollment

IITBombayX Data Analytics delivers enrollment activity data in a chart, and a report that you can view. Information about the computation reference has also been included.

7.1.2 Daily Student Enrollment Chart

The daily student enrollment chart is a stepped area chart: the filled area represents the total number of honor and verified enrolled learners on a particular date. For courses that offer more than one enrollment option or track, different colors represent the number of students who are enrolled with each option.

The chart currently includes these enrollment options and tracks, if they are offered for your course.

- **Honor Certificate**
- **Verified Certificate**

Moving your cursor over the chart shows a tool tip with the counts for each enrollment type - honor and verified.

The chart includes enrollment data for every day, beginning with the first enrollment (typically of the course creator). This data is also available for review in tabular format.

7.1.3 Enrollment Over Time Report

The daily total enrollment count, through the date of the last update, is available for review. Columns show each Date and its Total Enrollment for each mode.

7.1.4 Computation Reference

7.1.4.1 Enrollment Over Time chart

- **The filled area of this stepped area chart represents the total number of users enrolled in the course each day in each mode.**
- **The x-axis shows dates from course creation through the end of the last update period.**
- **The y-axis shows the total number of enrolled users.**
- **For courses that offer more than one enrollment option or certification track, different colors in the filled area represent the contribution of each option and track to the enrollment total.**
- **Each enrolled student is included in one, and only one, of the possible enrollment tracks on a given date.**

7.1.4.2 Enrollment Over Time Report

- **If a course offers students the option to pursue a verified certificate, the report includes columns for Verified Enrollment and Honor Code Enrollment.**
- **The Honor Code Enrollment column reports the count of students who opted to receive an honor code certificate for the course.**

7.2 Enrollment Demographics

Who is taking my course? Demographic data about your enrolled students helps quantify characteristics of the people who are taking your course.

IITBombayX Data Analytics delivers demographic data for three population characteristics: age, educational background, and gender. When students register, they can provide this information about themselves.

In IITBombayX Data Analytics, after you select Enrollment and then Demographics, you can choose Age, Education, or Gender to access a chart and reports to view.

7.3 Demographic Computations

During IITBombayX user account registration, students can provide demographic data about themselves. Demographic distributions are computed every day to reflect changes in course enrollment.

Students cannot change the selections that they make after registration is complete.

7.4 Age Demographics

How old are my students? Awareness of the ages reported by your students can help you understand whether a target audience is enrolled in your course.

7.4.1 Gaining Insight into Student Age

Students can report a year of birth when they register. Student ages, derived from year of birth, are provided in a chart and a report that you can view. Computation reference has also been included.

7.4.2 Self-Reported Student Age Chart

Each bar on this chart represents the total number of enrolled learners who are a given age, based on reported year of birth. Moving your cursor over a bar in the chart shows a tip with the number of students of that age.

The chart includes every reported age. This data is also available for review in tabular format. Note that students report ages of 0 and 100+

7.4.3 Age Breakdown Report

The number of students reporting each age, as of the date of the last update, is available for review. The report includes a row for each age, with columns for Number of Students and Percentage.

7.4.4 Computation Reference

7.4.4.1 Age Chart

- **Students can select a year of birth. Student age is computed as the difference between the current year and the selected year of birth.**
- **Each bar in the Column Chart represents the number of enrolled users (y-axis) of that age (x-axis).**
- **Students who did not provide a year of birth at registration are not represented in the Column Chart.**

7.5 Education Demographics

What educational background do my students have? Evaluating the stages of formal education that your students have completed can help you understand whether your course is enrolling people with the learning background that you expect.

7.5.1 Gaining Insight into Student Education

Students can select a âĂIJhighest level of education completedâĂİ when they register. Education data for the students enrolled in your course is provided in a chart and a report that you can view. Computation reference has also been included.

7.5.2 Self Reported Student Education Chart

The bars on this chart represent the percentage of enrolled learners who reported completion of a level of education. Moving your cursor over the chart shows the percentage for each level.

Depending on the goals of the course team, these distributions can be interpreted as indicators of the success of enrollment efforts, or indicate that changes may be needed to reach the target demographic.

Student education data is also available for review in tabular format.

7.5.3 Education Breakdown Report

The number of students reporting completion of each educational level, through the date of the last update, is available for review.

The report includes a row for each educational level and a column for the Number of Students and the percentage corresponding to each educational level.

7.5.4 Computation Reference

7.5.4.1 Educational Background Chart

- **Students can select a highest level of education completed.**
- **Each bar in the histogram represents the percentage of enrolled users (y-axis) who selected a completion level (x-axis).**
- **Percentages are calculated for the total number of students who reported an educational level, not from the total number of students enrolled in the course.**
- **The table that follows shows each IITBombayX label, the option that students can select at registration, and a brief description.**

7.6 Gender Demographics

What is the gender balance in my course? Knowing the male-female ratio in your course can help you understand who is enrolling in your course and whether the balance changes over time.

7.6.1 Gaining Insight into Student Gender

Students can identify themselves with a gender by selecting Female, Male, or Other when they register. Student gender data is provided in a chart and a report that you can view. Computation reference has also been included. Students who do do not provide this data upon registration are classified under "NOt Specified".

7.6.2 Self-Reported Student Gender Chart

The bars on this chart represent the most recently calculated percentage of enrolled learners who reported a gender of Female, Male, or Other. Moving your cursor over the chart shows the percentage for that gender.

Each of these course teams might use information about the percentages of enrolled men and women as a starting point for an investigation into how students learn about their course offering and make the decision to enroll in the course.

Student gender data is also available for review in tabular format.

7.6.3 Gender Breakdown Over Time Report

The daily total enrollment count with gender breakdown is available for review. Columns show each date, the total enrollment on that date, and breakdown columns for the number of people who reported each gender category and who did not provide this information at registration.

7.6.4 Computation Reference

7.6.4.1 Gender chart and report

- **Students can select a gender. The chart depicts the percentage of students who selected each choice (Female, Male, Other).**
- **The chart only includes students who reported their genders. The percentages shown in the chart are computed for the total number of students who did and who didn't select a gender.**
- **The report includes all enrolled students. For each day, the report includes the daily total enrollment count followed by columns that break down the total by Female, Male, Other or Not Specified.**

7.7 Enrollment Geography

Where are my students from? Enrollment geography data helps you understand the overall reach of your course.

7.7.1 Gaining Insight into Student Location

IITBombayX Data Analytics delivers data about student location in a map and a report that you can view. Computation reference has also been included.

7.7.2 Geographic Distribution Map

The map uses a color scale to indicate the percentage of total enrollment represented by students from each state. The darker the shade, the higher the enrollment percentage. You can view the enrollment total for each state: move your mouse over the map.

7.7.3 Geographic Breakdown Report

The columns in this report show each State and its Percentage and Total Enrollment.

7.7.4 Computation Reference

User location is determined without regard to a specific course. Users who are enrolled in more than one course are identified as being in the same location for all of their courses.

7.7.4.1 Geographic Distribution Map

- **The number of users and the percentage of the total enrollment is provided for each state.**
- **User data as entered at the time of registration for state is used for computation.**
- **The computational frequency and approaches used to determine user location and user enrollment status are different. As a result, you may note discrepancies between the total number of students reported by the Enrollment Activity and Enrollment Geography sections of IITBombayX Data Analytics.**

7.8 Student Engagement

To learn about what students are doing in your course, from the IITBombayX Data Analytics menu select Engagement. Select Content to investigate how many students are interacting with course content overall, and what they are doing. For data specifically about the navigation, select user navigation.

7.9 Engagement With Course Content

How many of the enrolled students are actually keeping up with the work? What are they doing? Content engagement data helps you monitor how many students are active in your course and what they are doing.

7.9.1 Gaining Insight Into Student Engagement

EdX Insights delivers data about student engagement in a chart and a report that you can view. Details follow. Computation reference has also been included.

7.9.2 Weekly Student Engagement Chart

The markers on this chart represent the number of unique students who interacted with course content. The graph plots three categories of engagement: an overall total for students who completed any type of course activity, and totals for students who clicked play for any course video and for students who submitted an answer for a problem. Each total is for activity completed within a one week period. To see the total count for each activity type for a given week, move your cursor over the chart to display a tip.

Activity is included beginning with the week in which the first page visit took place. The first page visit is typically by a member of the course staff shortly after course creation. This data is also available for review in tabular format.

7.9.3 Content Engagement Breakdown Report

The weekly breakdown of student engagement with course content is available for review.lumns show each Week Ending date and counts of active students, students who watched a video, and students who tried a problem.

7.9.4 User Navigation

This visualization shows the navigation of a particular user for a particular course. It shows the toal time spent by the user for different events along the course. The data has been fetched from the log files of the user using Big Data analytical tools (Spark). The events which have been captured from the log files are:

- **Courses These are log files events pertaining to miscellaneuos course interaction events.**
- **Navigation These are the events that of the log files corresponding to navigations, i.e. search throughout the course module.**
- **Problem These are the events captured from the log files pertaining to the problems either attempted, solved or accessed by the students.**
- **Video These are the events taken from the log files pertaining to video access events which include watching, seeking and navigating throyught the video.**
- **Discussion These are the events taken from the log files corresspond-ing to discussion forums.**
- **Enrollment These are the event taken from the log files pertaining to enrolling students.**

7.10 Student Performance

To assess how students are doing in your course, make a selection from the Performance menu. How are students answering questions?

7.10.1 Answer Distribution

This shows the learning pattern of students by their performance in the quizzes of the respective courses .For each course,we get a list of quizzes for that course and for each quiz we get a graph representation.

7.10.2 Answer Distribution Chart

Every Stepped Bar Chart in the graph represents the problems of the concerned quiz of the respective course .The Stepped Bar Chart shows the number of correct and incorrect responses for the selected problem.

7.11 Video Interaction

This visualization analyses the difficulty-level of any video of a particular course .The x-axis represents number of video-accesses for that video in the video-frames of 4 seconds and the y-axis represents the number of video-accesses for that video-frame.

The following are a few graphs that we have plotted:

7.12 Instructor Dashboard

The link made on the Instructor dashboard for OpenEdX Insight.The steps is as follows:

1. **By clicking on the Insight link on the Instructor Dashboard,the user gets redirected to the OpenEdX page.**
2. **Thus,the user can be able to see the OpenEdX Insight Dashboard on his/her screen.**

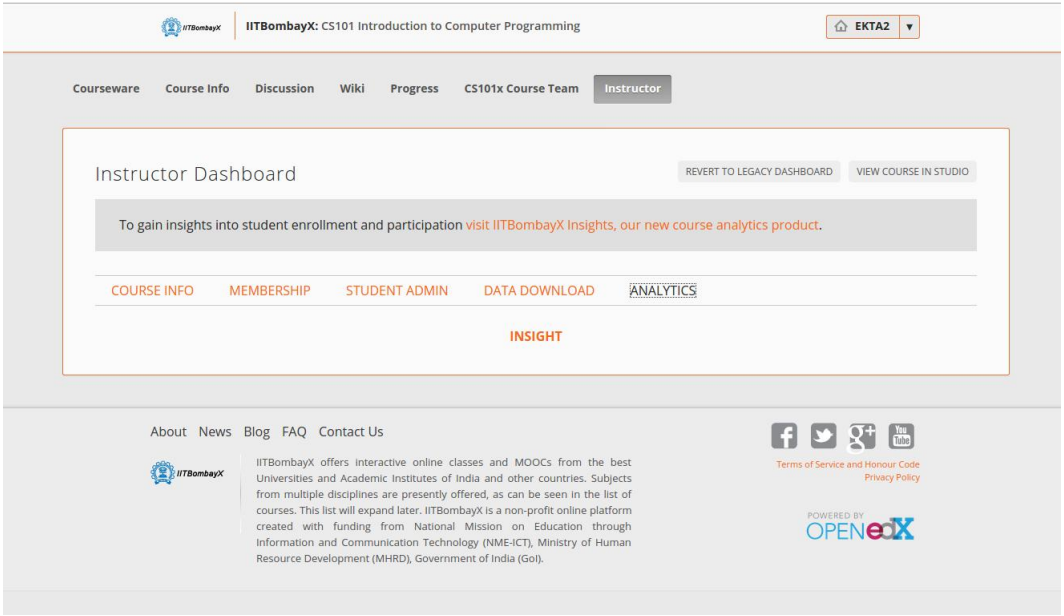


Figure 7.15: Instructor Dashboard

What required changes made to the Dashboard:

1. **Go to the following path and made required changes:**

```
edx/edx-platform/tree/master/lms/templates/instructor/instructor_dashboard_2/
CHANGES MADE:

<%page args="section_data"/>
<center>
<li="nav-item">
<a href="https://www.google.co.in" data-section="instructor_analytics" class=
```

2. **Template Description:**

```
course_info.html:
Consists of the following functions which is taken into consideration:
1. %if settings.FEATURES.get('DISPLAY_ANALYTICS_ENROLLMENTS')
2. %if settings.FEATURES.get('ENABLE_SYSADMIN_DASHBOARD', ''):
3. %if settings.FEATURES.get('ENABLE_INSTRUCTOR_BACKGROUND_TASKS'):

membership.html:
Consider the following link for the description of membership.html code:

https://github.com/edx/edx-platform/blob/master/lms/templates/instructor/inst

student_admin.html:
Consider the following link for the description of student_admin.html code:

https://github.com/edx/edx-platform/blob/master/lms/templates/instructor/inst
```

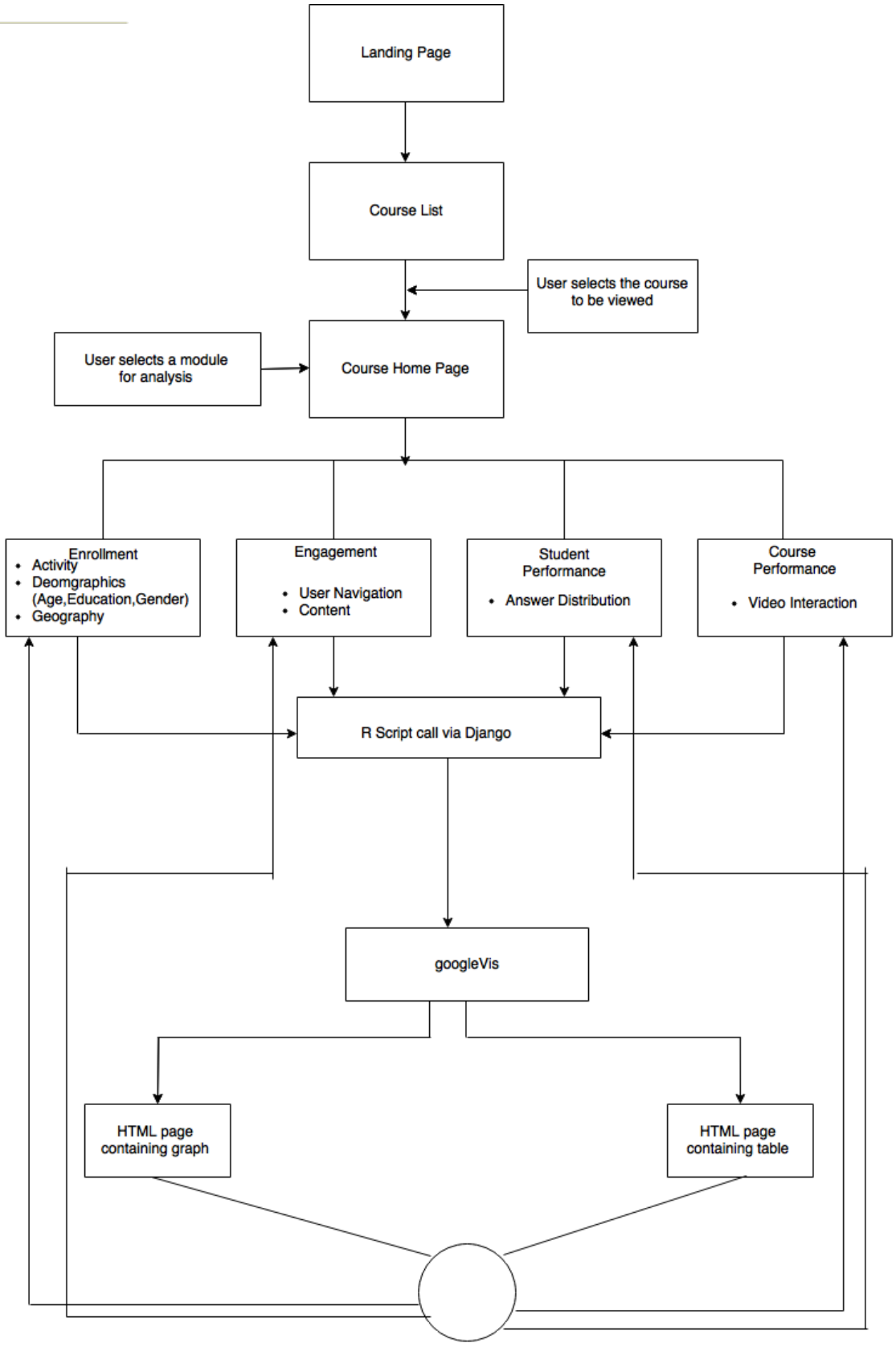


Figure 7.1: The Dashboard

Table 1.1: Educational Background chart

Label	Student Response	Description
None	None	No Formal Education
Primary	Elementary/primary school	Initial schooling lasting approximately six years
Middle	Junior secondary/junior high/middle school	Continuing basic education lasting two to three years
Secondary	Secondary/high school	More specialized preparation for continuing education or employment lasting three to four years
Associate	Associate degree	Completion of two years of post-secondary education
Bachelor's	Bachelor's degree	Completion of four years of post-secondary education
Master's	Master's or professional degree	Certification for advanced academic or occupationally specific education
Doctorate	Doctorate	Advanced qualification for original research

Figure 7.2:

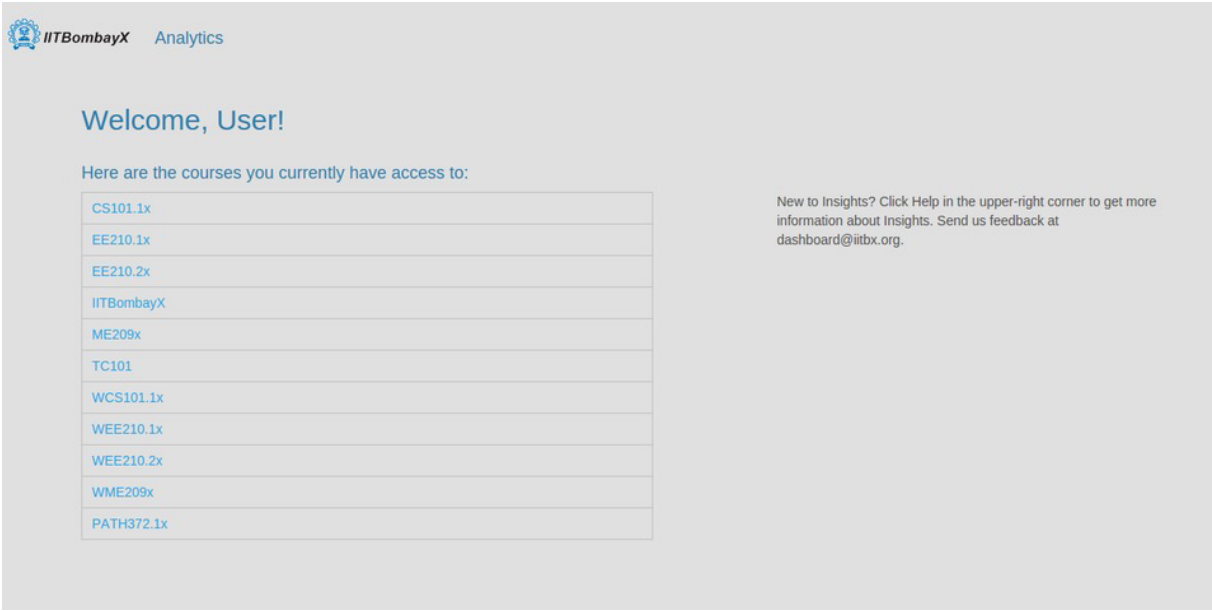


Figure 7.3: Course List Page

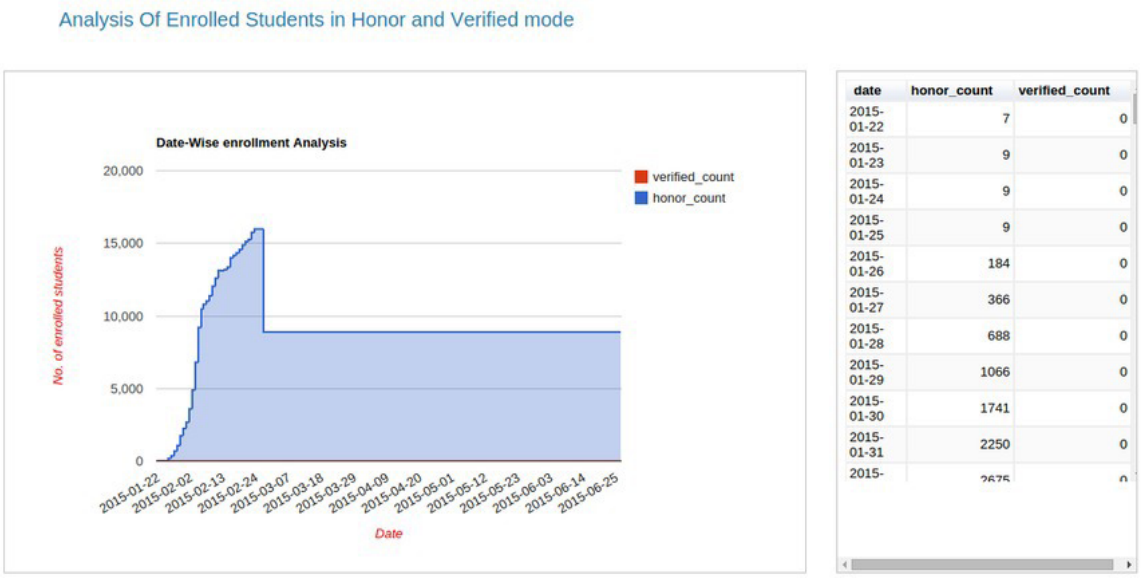


Figure 7.4: Graph showing number of students enrolled under honor and verified mode

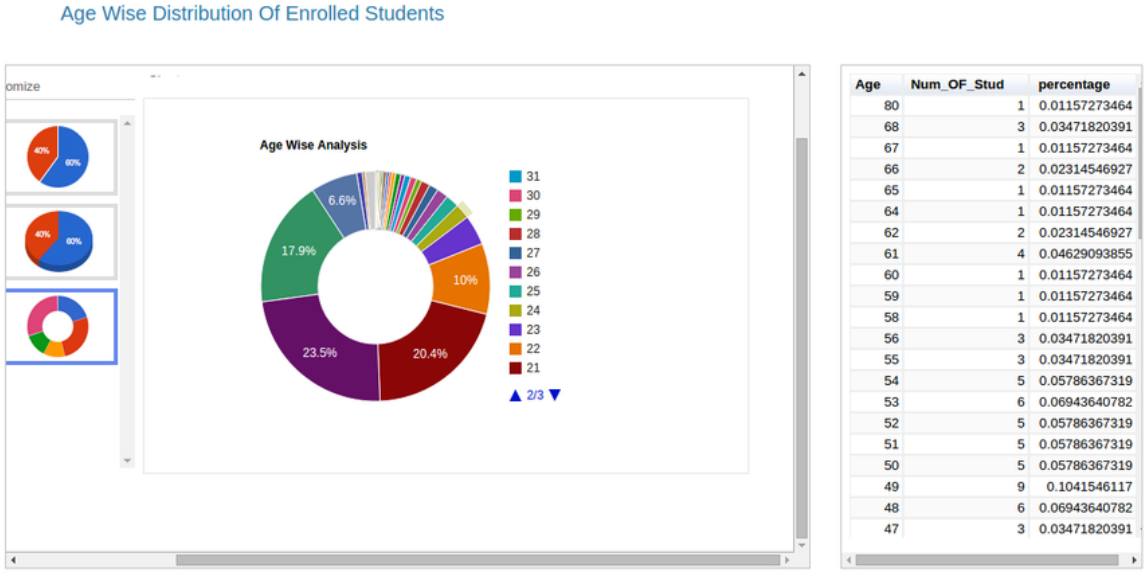
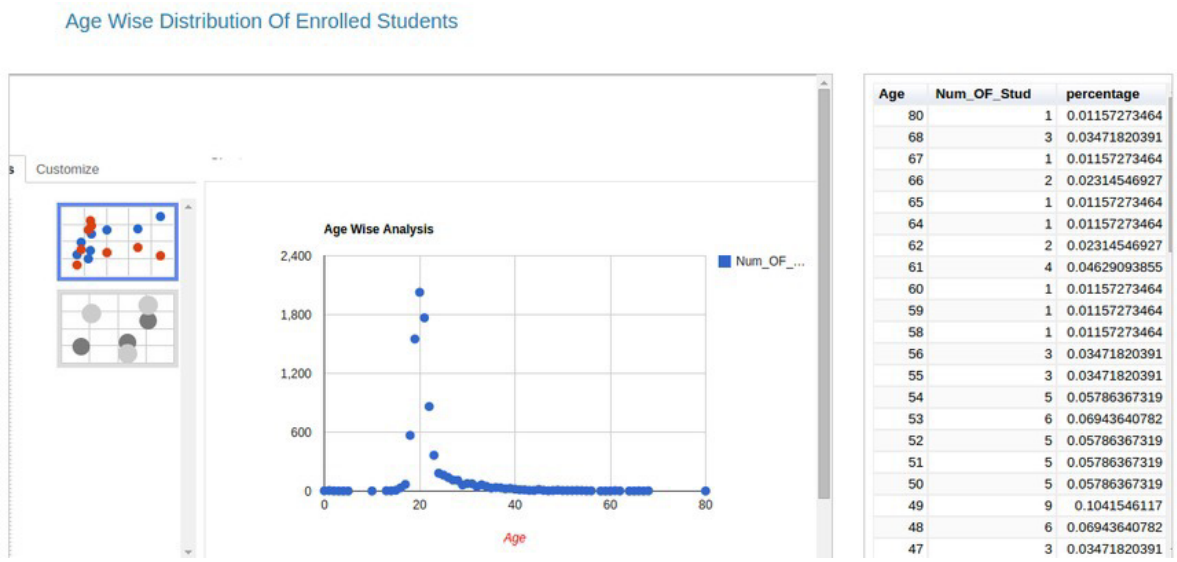
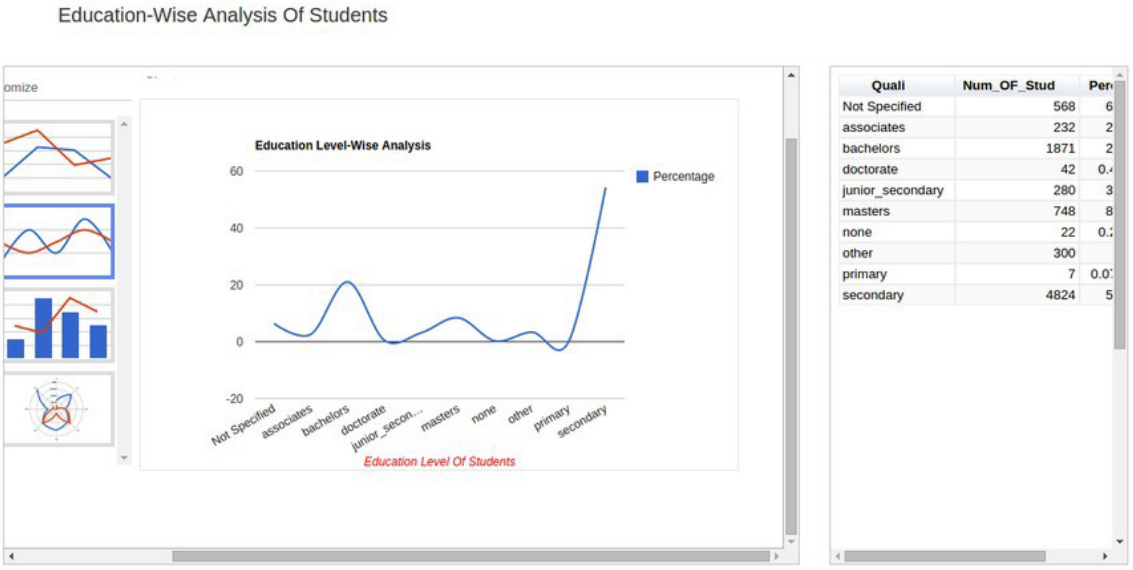


Figure 7.5: Graph showing age-wise student distribution



Age	Num_OF_Stud	percentage
80	1	0.01157273464
68	3	0.03471820391
67	1	0.01157273464
66	2	0.02314546927
65	1	0.01157273464
64	1	0.01157273464
62	2	0.02314546927
61	4	0.04629093855
60	1	0.01157273464
59	1	0.01157273464
58	1	0.01157273464
56	3	0.03471820391
55	3	0.03471820391
54	5	0.05786367319
53	6	0.06943640782
52	5	0.05786367319
51	5	0.05786367319
50	5	0.05786367319
49	9	0.1041546117
48	6	0.06943640782
47	3	0.03471820391

Figure 7.6: Graph showing age-wise student distribution



Quali	Num_OF_Stud	Per
Not Specified	568	6
associates	232	2
bachelors	1871	2
doctorate	42	0.4
junior_secondary	280	3
masters	748	8
none	22	0.2
other	300	
primary	7	0.0
secondary	4824	5

Figure 7.7: Graph showing student distribution based on education level

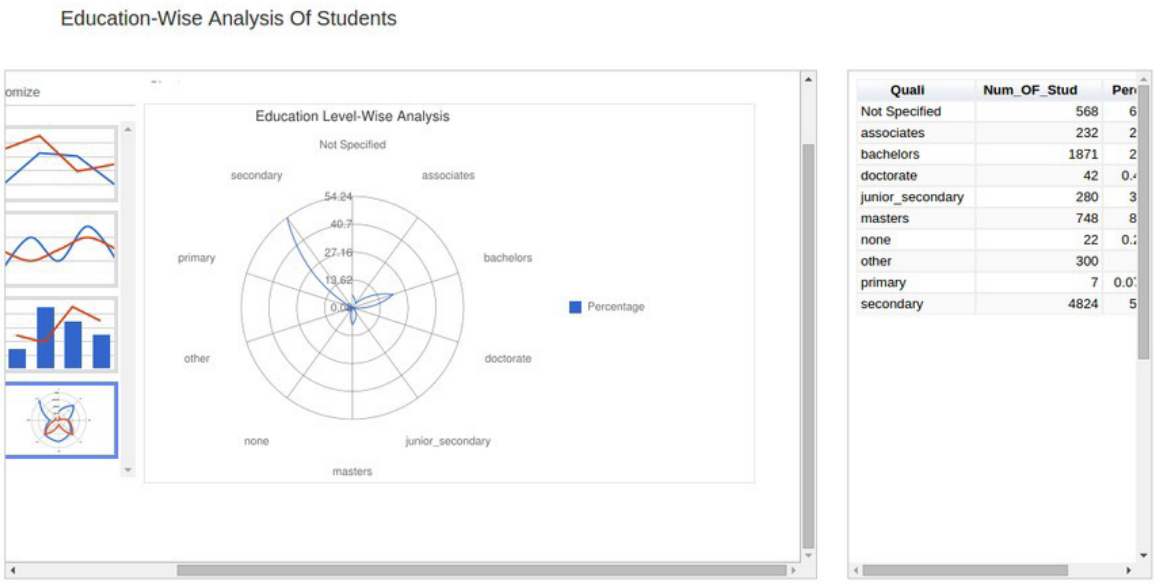


Figure 7.8: Graph showing student distribution based on eduation level

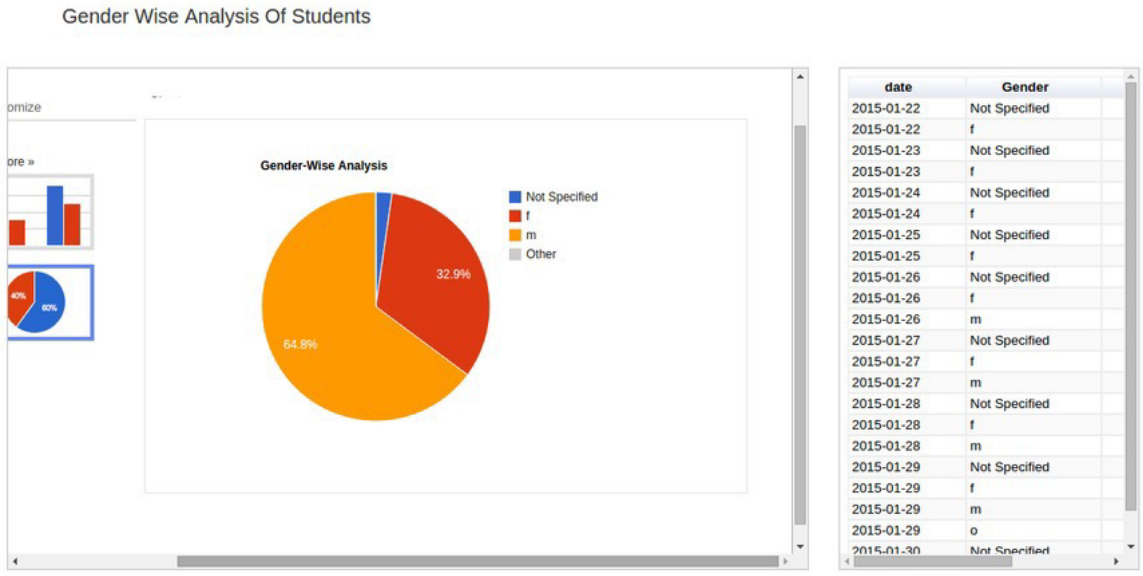


Figure 7.9: Graph showing gender-wise student distribution

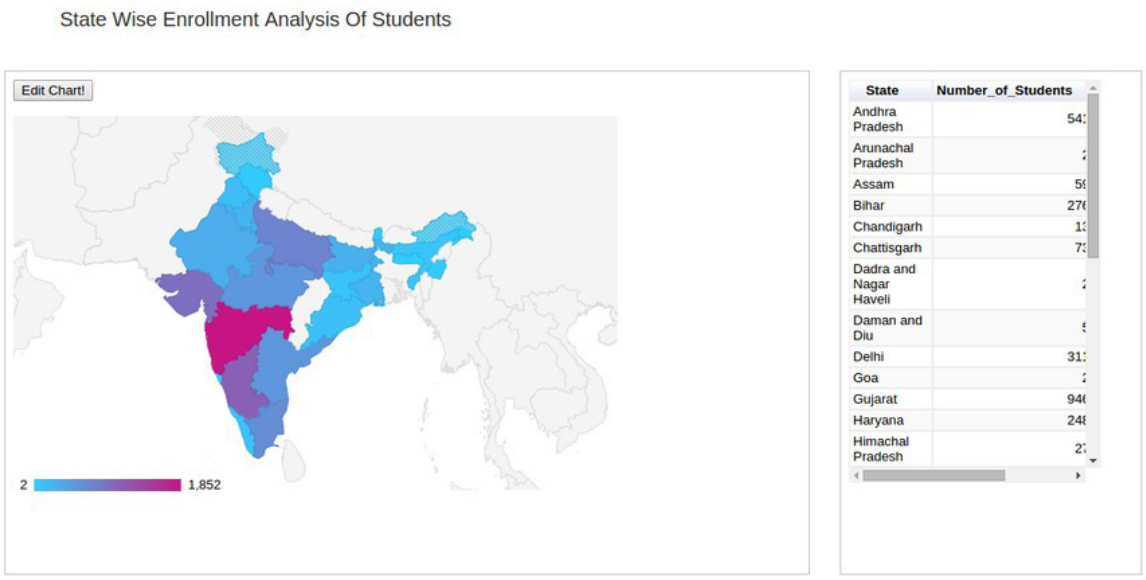


Figure 7.10: Graph showing geographical distribution based on student location

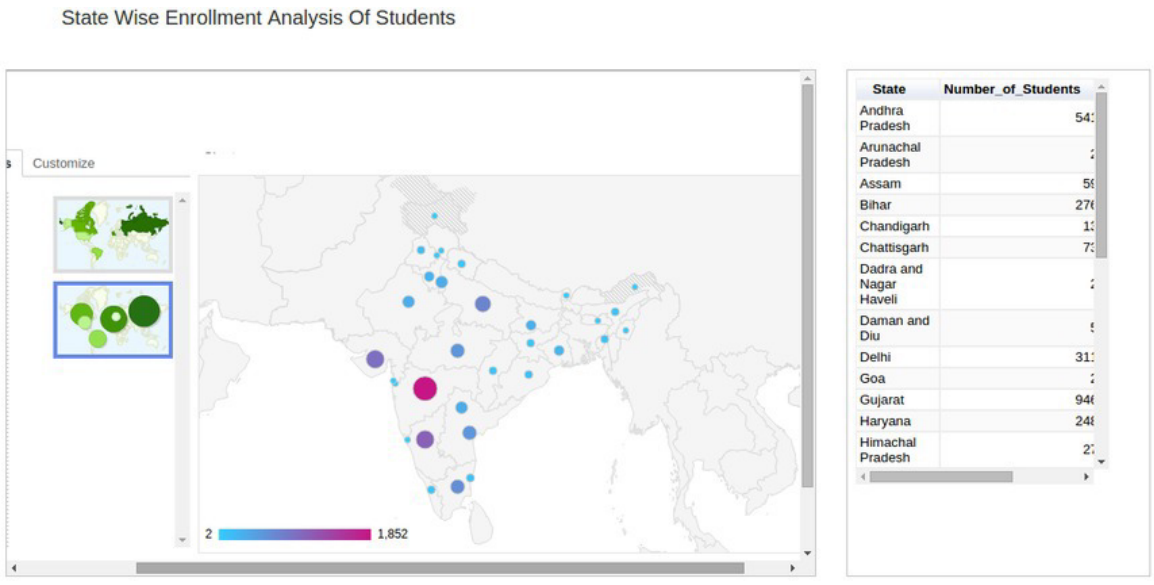


Figure 7.11: Graph showing geographical distribution based on student location

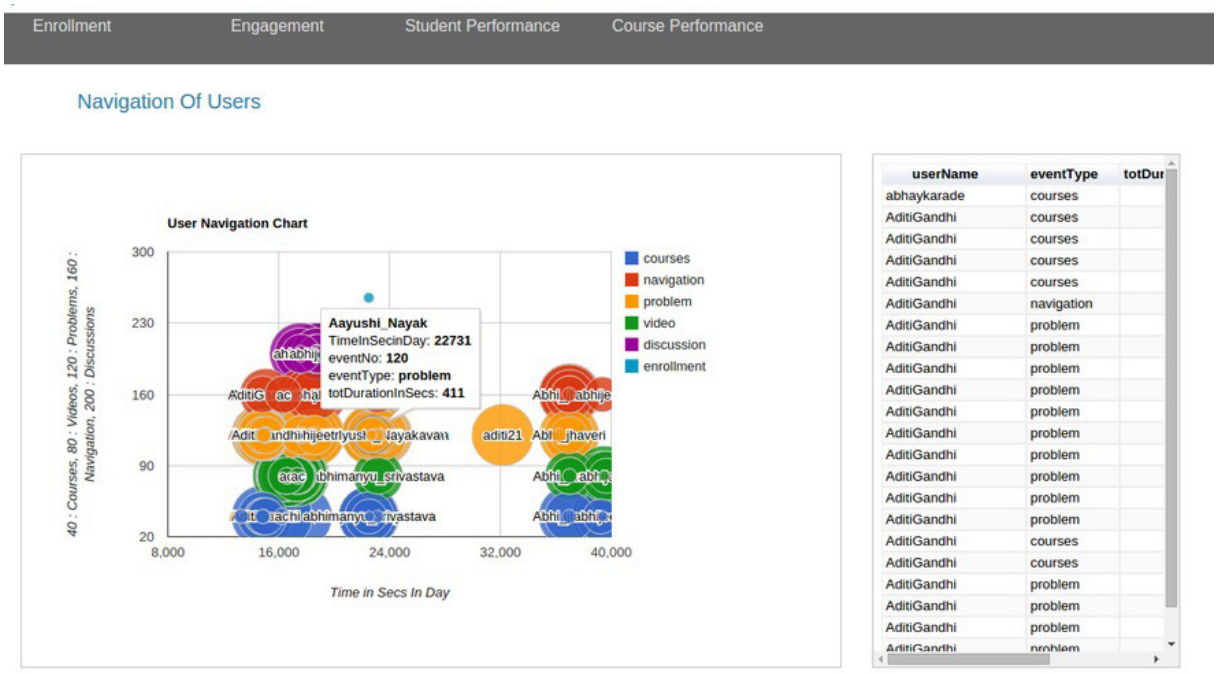


Figure 7.12: Graph showing User Navigation

Figure 7.13: Graph showing Student engagement - content

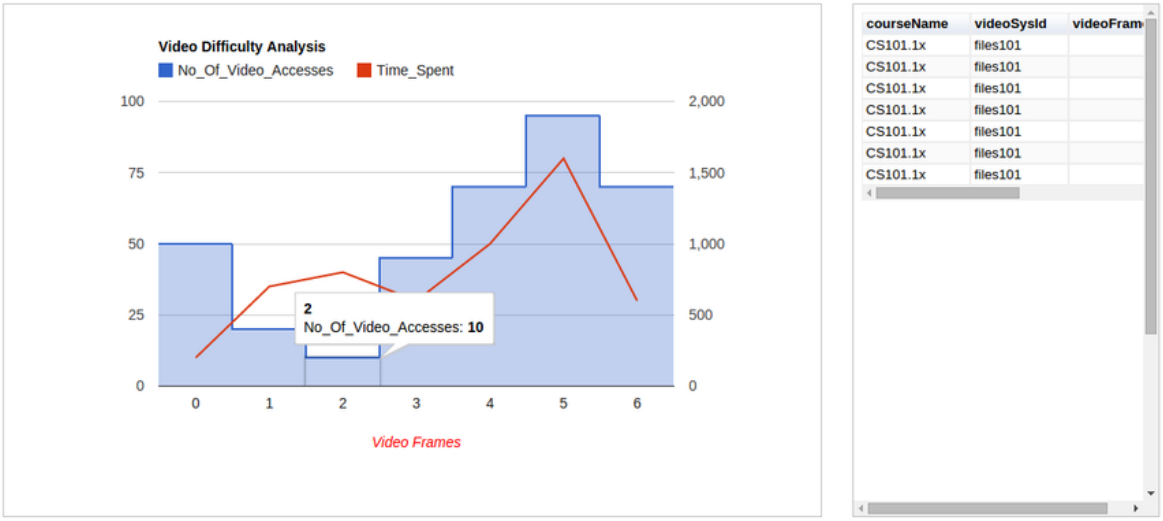


Figure 7.14: Graph showing Video access done by students

Chapter 8

Detection of difficulty areas in videos based on student activity

8.1 Introduction and outline of the chapter

This part of the project report talks about the project work done regarding the development of data models for the detection of difficulty regions in videos based on the students behaviour, recorded through the log events in IITBombayX. The chapter is outlined as follows. First, we introduce the problem statement that we wish to address through our work, and provide a basic outline of our idea to deal with the problem. Then, we talk about the steps that we have followed in order to create and formulate the data model. Finally, we describe how we had designed the data model and the functionality both at the processing and visualization level for a final implementation of the analytics module in the scenario of Big Data. After the final implementation and testing, the module was integrated with the OPENedX Insight analytics system for IITBombayX, and made available for use by the course instructors participating in the Blended MOOCs model. The entire work of this part was completed in 3 weeks of the internship.

8.2 Discussing the problem statement

IITBombayX is provides a number of Massively Open Online Courses (MOOCs) to be accessed by students from students all over India, irrespective of any barriers whatsoever. For the students participating in MOOCs, from the point of view of the course instructor, the primary mode imparting the lessons to the students, and the basic mode of communicating the knowledge to the students is through the lecture videos that the instructor publishes as part of the course. The videos are distributed among the various weeks and lessons in the courses, with each lesson having possibly multiple videos. Ideally these videos are watched by the students in succession, as the course progresses, and the student's

learning is successively evaluated through the tests and questions he has to complete online. Occasionally, a student may diverge from the normal train of events and watch the videos at his own pace, and out of order, but that is not our primary concern here.

Even though the student watches the videos online as part of the course, the student may (and they often do) face problems and difficulties in understanding certain concepts, much like in a classroom environment where the teachers deliver the lectures. In contrast to the classroom environment, here, the student can go back to a part of the lecture video over and over again, and watch and note the concerned part repeatedly until his or her doubt in the particular topic is cleared. While this is quite common, we have to note the causes for such occurrences. Among myriad other reasons why this could happen, we believe that the reason why students would have to repeatedly watch a certain section of the video is because:

- **The course videos are difficult for the students to understand.**
- **The course videos do not provide sufficient clarity on the subject that it deals with.**

Now, student behaviour such as this (namely, repeatedly watching certain sections of the videos) is a an information of interest to the course instructors managing and designing a particular course, as they would always be willing to improve upon the course and help the students in their learning process. If this data were made available to the instructor in an understandable interface, then the instructor would be able to tell whether the the participants in his MOOC are indeed finding the videos in the course useful for their purpose, or are simply spending their time perplexed over certain regions of the videos and are thus not being able utilize the learning advantages of the MOOC system - missing the forest for the trees, as it were, and ending up learning little, none, or worse - going back with mistaken understandings regarding the subject of the course. Thus, we understand how important it is in order to address this issue - and we intend to solve this through data analytics tools and data modelling.

8.3 Basic outline of the idea

The solution that we came up with was based on the fact that the OPENedX InSight platform, on which IITBombayX is based, records all the student activities on the platform as the logs. In other words, everything that the students do on the platform while accessing and interacting with the course content - including video actions, are recorded by OPENedX, and made available to all those who need to use it. We can also identify not only the users responsible for an acitivty, but also

the particular course module (problem, question, page or video) that the user interacts with. In this way, we can also track the behaviour and activities of a student with respect to a particular video in a course, and the various video events such as pausing a video, playing it, seeking to a certain part in it, and so on. These events can then be analysed to determine meaning and inferences regarding the behaviour of a student on a particular video.

Primarily, we could focus on only the events such as pausing a video, playing a video, or seeking back and forth on a certain region of the video, and collect the locations of the pauses and seeks in the video to determine how much of the video has been watched by the user, and how have the different parts of the video been watched by the user. Once these events are collected, and the data processed, appropriate visualizations - such as graphs, charts, or some other means - could be developed on this data, or tools and alerting systems developed in order to tell the instructors of the course about the different regions in the video where the students have indeed faced difficulties. An alerting system would require the system itself to be able to compare the data of different videos within the same course, undertaken by the same instructor, to be compared to each other and then determine from among them which are the regions of the videos where the students are facing difficulty in understanding.

Once such regions of the videos have been detected and the alerts issued or visualizations provided - the instructor, knowing the subject matter that corresponds to the regions of difficulty, would then take appropriate measures on his side to make the course contents more lucid for the students participating in his course. Solutions to this problem by the instructor could range from the following:

- **Adding more explanatory material, such as text materials, links, PDFs, and so one - or similar materials,**
- **Release of an additional video explaining the matter in a more lucid fashion for the students; covering materials which would enhance the understanding of particular subject matter by the students; or even the re-release of the same video, with edits and additional content for explaining the subject relating to regions of difficulty to the students,**
- **More quizzes and practice exercises, so that the students may be able to learn better by progressing through the matter using practical examples and hands-on experiences of the problems, answering questions, and going through corrections based on evaluations of their performances,**

And so on and so forth, as the instructor may see fit for the situation.

Our project mainly aims to solve the problem mentioned in the previous section through the development of a data analytics module as a part of the IITBombayX InSight data analytics system. The broad functioning of the system would be as follows. The system would periodically go through the log data generated for the platform for every single video, assess the behavioural trends and activities of the students, and process it to determine the difficult areas in the corresponding videos on the basis of different parameters such as the amount of views for the different regions of the videos, time spent, etc. Once the processing of the data is completed, we would have the analysed summary data which could then be used for drawing inferences, learning, or visualizations for the analytics dashboards. One approach is to directly visualize the data on the InSight dahsboard interface provided to the instructor for a course using the summary data for a video directly. Alternatively, we could determine the regions of difficulty in the videos and compare these performances over all of the videos, and determine a trend of watching the videos; based on which, it would be able to tell, or classify whether a video has been deemed by the students as too difficult or just perfect for the students. In the current project however, we have focused chiefly on the process of generating the summary data, starting from the log data, and visualizing it at the end on the OPENedX InSight dashboard, allowing the instructor to make intuitive decisions based on the graphs plotted based on the data showing the metrics developed on the video events.

8.4 The Project Flow

In this section of the report, the details of the work done to achieve the final application are described. Starting with the raw data in the form of the log files, a considerable amount of effort has gone into several steps involved that transform the raw, uncleaned and unintelligible data into cleaned, well formatted data, to processed summary data and finally, its visualization on the dashboard. The steps can be clearly outlined as follows:

- **Extracting the data: This step involves the extraction of the data from the huge amounts of the log data that is generated by the IITBombayX platform every day, periodically, followed by converting it into a format and saving it on a store which can be regularly and efficiently accessed for processing by the further stages of the applications.**
- **Processing the data: This step involves the accessing of the required data from the saved stores in an efficient manner and properly process it and clean it prior to feeding it to the data model. It will be explained later in detail, but it so happens that even after retrieving the data from the stores, it is required to do certain processing on**

the data before it can be given to the data model for generating the relevant feature data from the events.

- **Creation and prototyping of the data model: Following the extraction and processing of the data, a data model is used for extracting the feature data that we would be using for our summaries. In this stage, a working prototype of the data model was developed to work on a sample of real-world data, so that it could be developed, revised and corrected as fast as possible, and then taken to the final implementation. The prototype development stage involved the use of MySQL and Python scripts and libraries. the extraction of the required data was done from MySQL tables using carefully crafted SQL queries. The data cleaning and processing code, and the code to implement the prototype of the data model to generate summary data, was done using a Python program.**
- **Final implementation of the data model: The working prototype of the model was then scaled up and coded for frameworks to work in a Big Data environment, where the data model and the processing tasks would be working on log data generated in a scale of potentially tens of gigabytes per day. This stage involved the final implementation of the data model and its deployment in the IITBombayX InSights dashboard, including appropriate visualizations to aid the monitoring and decision making process of the course instructors. The Big Data technologies used to achieve the final implementation include the Hadoop Distributed FileSystem, Hive data warehouses (which use Hadoop for storing the databases and tables created and accessed in Hive), and SparkSQL to provide a fast and very efficient method process the massive amounts of data using a MapReduce workflow and parallelized functions.A visualization for the final summary data, which determines the total number of accesses and total time spent in the different regions of the video by the people who have seen the video, is made available on the front end — which use Django framework, and R visualization scripts using GoogleVis libraries.**

8.4.1 Source of data for the model

To begin with, the prototyping of the data model or even the creation of the basic application required for us to have some real data to work on, giving us the details of the student's activities on the IITBombayX platform. Now, the OPENedX platform generates the log data regarding each and every student activity, and records it in log files, which are stored on the servers. Typically, these log files span over several gigabytes in size, and therefore, any processing on these would be require application of Big Data tools. The individual logs themselves are recorded as JSON strings. A JSON string consists of several key-value pairs, and a single key itself may lead to a nested structure having more key-value pairs. A sample of the log event looks as shown 8.1:

```
{"username": "arinjoy15", "host": "10.105.25.74",
 "event_source": "server", "event_type": "/dashboard",
 "context": {"user_id": 11, "org_id": "", "course_id": "",
 "path": "/dashboard"}, "time": "2015-06-12T10:20:08.703318+00:00",
 "ip": "10.105.1.7", "event": "{\"POST\": {}, \"GET\": {}}",
 "agent": "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:38.0) Gecko/20100101 Firefox/38.0",
 "page": null}
```


Figure 8.1: A sample log event generated .

Now, the accessing the logs themselves in such a manner is a difficult and tedious task. The structures of the JSON strings is not the same for all events, and while all events have certain common keys, they also have certain key-value pairs that are descriptive of the particular events themselves. For example, a log event describing a video event would have keys for the users, time of event, event-source, and so on - so would an event that corresponds to a student-dashboard access. However, the video event would also contain details of the video event, the video id, the current location of play or pause, and the current video speed; which would not be there in a dashboard access event. Thus, we explain the difficulty of accessing these events in their raw form.

To solve this problem, the log events were flattened out, to create tuples having fields that would have data according to the type of the event in question. So, fields that do not relate to the data for a particular event would be left NULL, while the other fields related to the same would be having valid data. These flattened out logs were then put in the MySQL table corresponding to the User logs in the Data Analytics database, so that they could be accessed and queried as required through simple MySQL queries. This data served as the source data for my task. There were certain errors and discrepancies in the database tables created from the logs, which are discussed in the subsection on 'Extracting and Cleaning' the data.

8.4.2 Processing the data

In order for us to determine the model on which we were to base our system, we drew our data from the IITBombayX log files, which were cleaned and made appropriately accessible through MySQL queries. Now prior to actual processing of the log data for the videos and with respect to the user activities, the data needs to under go further extraction, cleaning and pre-processing - as the log data also contained several redundancies and errors. All of these tasks were done at this stage of the project. This step also required us to understand the nature of the logs at a fundamental level, since in the following stages, we would have to design our data models to work on the processed and filtered events. A description of the activities undertaken in this stage are in the following subsections.

8.4.2.1 Extracting and Cleaning the required data

A first step involved exploration of the log events and the associated attributes of the logs, to identify the features that we would require to identify each video element for each course, each video event, and to extract the data. These are the features that we would later use for the creation and framing of our data model. At this stage, SQL queries were used to filter and select the relevant data for each video and each user watching it. The tables that we used in this stage for our queries included the following:

- **UserSessionOldLog : The MySQL table which contained the events from the user logs put into a single table, in a flattened format. The table had a total of 58 columns, and contained nearly 4 million entries. However, upon exploration of the values of the attribute using specialized queries, the attributes we found to be of interest to us were as follows:**
  - **userName : The name of the user who is responsible for the event in the user log,**
  - **moduleSysName : The system id of the module (in this case, usually the video) which is being used or accessed by the user concerned.**
  - **courseName : The name of the course in which the module is located. This allows us to detect the videos corresponding to a particular course while creating summary data for a video.**
  - **currVideoTime : The current location of the video when the log event was recorded.**
  - **oldVideoTime : The old location of the video when the log was recorded (required for certain video events such as seeking),**
  - **createDateTime : The system (real-world) time when the log event was created by the system - allowing us to arrange the events in a chronological fashion to determine the temporal sequences and semantics of the events generated.**
- **CourseVideos : This is the MySQL table which contained all the details of all the videos in the different courses that were available on the IITBombayX platform courses. The table itself contained 14 attributes, and contained 1628 entries. Upon exploration of this MySQL table too, we found that there were only certain attributes of interest to us, that were as follows:**
  - **videoSysId : Similar to UserSessionOldLog.moduleSysId, this is also a system id which is used to determine the individual video modules that are begin referenced in each event or even each activity.**
  - **videoUTubeId : This field is a string field containing the YouTube id of the corresponding lecture video whose videoSysId is provided; the videos are actually published on Youtube, and a link**

to these videos is provided in the corresponding units and sections in the actual course.

* videolength (added) : This is an additional field which was computed and added to the fields later by work done as part of this project. This field would contain the floating-point values indicating the length of the YouTube videos (i.e., the lecture videos) obtained from the YouTube attributes of the videos. The process of extracting this data is elaborated in a later section entitled 'Video length extraction using Google YouTube API'.

One point that needs to be noted here is that while the code had been written to flatten out and clean the log JSON strings, it was not complete in extracting all the log information correctly from the logs. A number of erroneous situations were encountered while going through the tables UserSessionOldLog and CourseVideos. For example:

- There was one CourseVideos entry (id='2017625218ba4309b4cd42309f5d82e2')which did not have an YouTube id, chapter title and which meant that no information about the video could be extracted from the YouTube API even if we wanted to.
- In certain cases, the events such as save_user_state were supposed to record values in the fields for oldVideoTime and currVideoTime, but instead, they showed up to be NULL. This may have happened in the log files due to a network error, due to which erroneous data was recorded in the files, or due other factors beyond our control at this point.
- For all the save_user_state events, despite clearly being video events associated with a definitely identifiable video, the attribute moduleSysId did not contain the videoSysId of the corresponding video in CourseVideos. It was NULL for such cases, making it impossible to identify which video the event was associated with. This was a problem at the time of working on the project, but which was subsequently fixed in the course of the project.
- The sequence of events that was observed on practice simulations of the platform was not necessarily observed in the log data collected for the events. In many cases, the events were recorded out of order — and hence required inference rules to associate them correctly to avoid errors. Again, this could be due to network errors, or some erroneous situations or faults that cannot be controlled or monitored by us at this point.

All of these errors, nevertheless, were dealt with using appropriate generalization cases and writing the codes and conditions for them accordingly. These are discussed later in these section, under 'Redundancy removal and Error correction'.

Now, in order to start working on the prototype, we needed to find the most efficient and complete queries that would extract all the necessary data that we would require for our data model. As far as we were concerned, we simply needed to select for a video, and for a user who had watched the video, to get all the events of that user accessing that video. This could then be processed upon by our programs.

To start with and develop our data model and our queries, we first focused our search on videos belonging only to the CS101.1x course. This course alone had 155 videos, out of the nearly 1600 videos of all the courses that are currently provided on the platform. In order to further restrict our search for designing the model, we selected the video with the most number of views, based on the log data, and then selected the user who has accessed that video the most number of times. The end result of this process was a sufficient amount of log data for a single user viewing a single video of a particular course, which would provide enough insight into the patterns of general behaviour of a student or learner.

Some of the MySQL queries that we actually used for our data model prototype are as follows:

- **Fetching top 50 most watched videos in CS101.1x**

```
select t.*, CV.videoUTubeId, CV.videolength
from (
select moduleSysName, count(eventName) eventCount
from UserSessionOldLog
where courseName='CS101.1x'
and moduleSysName is not NULL
and eventType='video'
group by moduleSysName order by eventCount desc limit 50
) t,
(
select * from CourseVideos where courseName='CS101.1x'
) CV
where t.moduleSysName = CV.videoSysName
order by eventCount desc;
```

- **Highest watched video : videoSysId '67a8559582864d6a8148e2ef5c997e8f'; So, number of viewers in descending order of activity were found as follows:**

```
select userName, count(eventName) watched
from UserSessionOldLog
where courseName='CS101.1x' and
moduleSysName='67a8559582864d6a8148e2ef5c997e8f'
group by userName order by watched desc limit 10;
\end{tiny}
```

– **Finally, for a given user, we found out the distinct events of interest giving us the behaviour of the user in the following manner (say, for the same video as before, but userName='ricky'):**

```
select distinct eventType, eventName, moduleSysName, eventSource,
oldVideoTime, currVideoTime, createDateTime
from UserSessionOldLog
where userName='ricky' and (eventType in ('video')
and moduleSysName='67a8559582864d6a8148e2ef5c997e8f'
or eventType in ('video', 'navigation') and
eventName in ('pageclose', 'saveuserstate') )
order by createDateTime;
```

8.4.2.2 Video length extraction using Google YouTube API

In order to determine a way to measure the relative amount of time spent by the student on a particular video (such as the fraction of the video playing time), we needed to have an idea of how long the video actually is. A student may spend the same amount of time on two different videos, but the different lengths of the videos in relation to the time spent on the videos would be a clear indication of actually how much attention and dedicated effort the student is providing to the video in the course. A higher percentage of time would mean a more engrossed attention, while a lower percentage would probably mean a casual learner.

However, this data regrading the duration of each of the YouTube videos was not available to us in the first place, as the table CourseVideos did not contain any field to determine the length of the videos . So, an additional work was done to extract the actual duration of the videos on the all courses using the YouTube id's available for the videos in the CourseVideos table (i.e., the videoUTubeId attribute). The lengths of the videos extracted from the JSON style strings returned by the Google APIs were processed into float values, and the result stored in a new column in the tables, called 'videolength', through a Python script.

It should be noted here that the Google API requires an authorisation key to be supplied by the developer or the program in order to access the API. For the script used, I have hard-coded a developer key I obtained from the service, and it can be used effectively, subject to certain daily quota restrictions (50,000,000 calls per day, 3000 calls/second/user).

The Google YouTube API basically sends GET requests online to the API together with the Youtube video id and the fields of the 'video' element that it has to return. For our case, we required only the contentDetails attribute of the video element, in which, the value corresponding

to the key 'duration' gave us the duration of the video in a ISO 8601 format (PT59M59S). This was processed and converted to a float value, which was then stored in the MySQL table through the script using a MySQL connector for Python.

The following is the code for the same:

```
#Proxy settings - change accordingly
import os
os.environ['http_proxy']=
     'http://arinjoy15:arinjoy15*@proxy.cse.iitb.ac.in:80'
os.environ['https_proxy']=
     'http://arinjoy15:arinjoy15*@proxy.cse.iitb.ac.in:80'

from apiclient.discovery import build
from apiclient.errors import HttpError
import mysql.connector

#creating connection and obtaining cursor
cnx = mysql.connector.connect(user='root', password='',
          host='127.0.0.1', database='IITBxDataAnalytics')
cursor = cnx.cursor()

#accessing list of video id's from the table.
query =
("SELECT videoUTubeId FROM CourseVideos WHERE videoUTubeId IS NOT NULL")

cursor.execute(query);
videoslist=dict()

for video in cursor:
videoslist.setdefault(str(video[0]),0.0)

#creating Youtube API service through the key
api_key = INSERT_OWN_DEVELOPER'S_KEY_HERE
youtube = build("youtube", "v3", developerKey=api_key)

#processing for each video
for video in videoslist.keys():
timestring =
    str(youtube.videos()
    .list(id=video, part="contentDetails,snippet")
    .execute().get("items",[])[0]["contentDetails"]["duration"])
    .split("PT")[1]
timestring = timestring
               .replace("H",":").replace("M",":").replace("S",":")
timestring = timestring.split(":")
```

```
second=0
minute=0

if not timestring[1]=="":
second = int(timestring[1])
minute = int(timestring[0])
videoslist[video] = minute*60.0 + second

#updating the table columns
for videoName in videoslist.keys():
query=("UPDATE CourseVideos SET videolength=%s WHERE videoUTubeId=%s")
    #to change to include the list of video id's to process
    #in a single query
data = ( str(videoslist[videoName]), str(videoName) )
cursor.execute(query, data)

cnx.commit()
#closing the cursor
cursor.close()
cnx.close()
```

It should be noted at this stage that in the table for the course videos, there were certain errors int e details of the videos recorded. In particular, the video with the id '2017625218ba4309b4cd42309f5d82e2' had neither youtube id nor a title name. Hence this video had no duration calculated from it, as it was evicted from the result set by the IS NOT NULL condition in the WHERE clause.

8.4.2.3 Redundancy removal and Error correction

In this stage of processing the data prior to applying to the data model, we worked on removing the errors, redundancies and reordering of events in the logs. This was necessary because based the log data that we received, even if they were flattened out, the simple SQL queries would not be sufficient enough to get the precisely clean and concise data that we would require for the data model. There were many instances of the same events that were recorded multiple times at the same instant, several instances of the same event recorded over consecutive times, erroneous occurrences of events recorded at the wrong times, and several other errors. So, we approached this problem as follows: From the list of events, keep only those events that corresponded well with the corresponding video watched by the user, and eliminate the rest of the events as they were redundant or wrong, or just useless in our current context. Doing the operations specified in the following lines not only removed the wrong and unnecessary events, but also increased processing time in view of the fact that the number of distinct events that needed to be

processed were reduced by a large factor.

Firstly, in order to remove multiple occurrences of the same data (for example, log entries having the same eventType, eventName, video module, and positions of timing, and even the time of recording the video), we introduced the DISTINCT clause into the MySQL queries to remove such occurrences. We found upon comparing that the use of the DISTINCT clause alone reduced the number of necessary and important events by a very large number. Also, the query ascertained that the events returned were arranged in the chronologically increasing order of time, so that the events could be processed in the order in which they actually occurred for the particular video and the user, allowing correct processing for our data model program.

However, this as far as the error correction and redundancy reduction by queries goes. Once these data were returned as lists of events by the queries run on the mysql connector functions (cursor.execute()), the rest of the error detection and correction were done as part of the Python programs written for the data model, although, the data model and the error removal scripts are separated from each other. The program was written in order to eliminate such rows, instead of using a join operation, because the join operation would take a longer amount of time (given the large number of events as entries in the table, requiring a self-join, therefore), and furthermore, it could be afforded, as the data was only required for the video once in a specified period (generally, a day), and so this computation could be done without much hassle.

Following the entries returned by the MySQL queries, we proceeded to investigate into the results that were returned by the queries. It was found that despite the fact that the events returned by the query were all distinct with respect to the attributes being selected by the SELECT operator, there were several occurrences where the same event was being recorded multiple times, but over different times (over successive values of the createDateTime attribute, separated by say, 1 second in real world time). In such cases, we would consider only one occurrence of such an event, by comparing these events pairwise in sequential order. So, if for two successive events, all the attributes fetched had the same value, except for the createDateTime attribute, we kept only one of them in the new list which is being prepared and the repetitive instances were ignored.

```
if not  (( activity1.eventType == activity2.eventType ) and
         ( activity1.eventName == activity2.eventName ) and
         ( activity1.oldVideoTime == activity2.oldVideoTime )
        and  ( activity1.currVideoTime == activity2.currVideoTime ))
{
     Then keep the video in the final list of events.
```

```
}
else Discard
```

Now, our simple query also returned events such as page_close and save_user_state for events which did not correspond to the current video under consideration. However, such specific requirements for returning the rows/entries could not be specified in the queries, and we thought it better to process it through the Python program itself. So, we came up with the following programmatic solution: Keep ignoring events until the first event for the video under consideration is detected, and then apply the necessary filters to include events for the videos (as mentioned before), until the page_close event for the video occurs, which is the last event for a video under consideration. Once a pageclose event occurs, include the most relevant save_user_state event for that page_close event, and wait till the next relevant event for the video under question. Also, the data model in our design required that the page_close event be followed by the most relevant save_user_state event (in case multiple save_user_state events were recorded for the video in the list of events), because ideally, each page_close event would be followed by a save_user_state event. The logic for doing so is discussed in the next section where we talk about the general flow of events while a video is being played. Thus, all the events for a video being played are covered.

Furthermore, if there were such events for the current video where the video positions given were wrong (such as, being out of the duration of the videos), then these events would be evicted from the list immediately. There were also certain blatantly wrong instances of user log events, where certain necessary and characteristic fields were left NULL completely. Such cases of events too were evicted from the lists immediately.

Following this stage, we were able to get a cleaned, concise view of the activity of users for that video. This usually contained a much more manageable number of rows than the original data, which could be efficiently processed via a program as well as examined manually. We proceeded to design an algorithm to implement the criteria to extract this data from the log files, examined in order of the createDateTime field in the tuples. We aimed to cover all the tuples in the result in a single traversal, and calculate the data for the different features for the video.

8.4.3 Creation and prototyping of the data module

In this part of the activities, we analysed the various video events and determined a model to extract the features we will use to determine

the time spent by the students on the videos they have watched, and accordingly use it for our summary and visualizations. We had decided on two features that we would map in order to find out the viewing activity of the students with respect to a particular video, which are as follows:

1. **Count of the number of times the different regions of the videos were accessed by the different students.**
2. **Total duration spent on a particular region of the video by the student.**

The hypotheses that formed the basis for our selecting these two features is as follows:

> ***The more number of times a particular region is visited by students, and the longer time students spend there, the more difficult it is for them.***

This hypotheses is justified because if the students repeatedly access a particular region of the video, and go through to actually watching it, then there would be an increase in not only the time spent by the user in that region, but also the number of accesses to the particular video section. In the worst case - the section might be the entire video. This would happen if the student was indeed stuck/perplexed by a particular topic discussed in the video.

Now, in order to find out the given metrics for a particular region of the video, it is essential to divide the video into several parts in the first place. For this, we had considered the concept of a timeFrame to divide the different regions of the video. A timeFrame is essentially a region of the video, x seconds wide, which is accessed by a user whenever he watches through the part of the video being spanned by that timeFrame, or completes more than half of that time frame during one play-pause session. So basically, we considered each video to be divided into a series of timeFrames of a specific width, and computed our metrics for assessing the difficulty regions by counting the number of accesses to the different timeFrames, and the total times spent by a user (or multiple users) in a particular timeFrame region. For the purpose of determining the timeFrames too, we would definitely need the video durations for each video, and this was also another reason why we proceeded to retrieve the video lengths from the YouTube data regarding the videos obtained using their YouTube id's.

In order to give an outline of analysing the log events, we must first discuss how the video events are actually appearing in the log files generated by IITBombayX. In order to truly understand this entire process, simulations and experiments were done by hand using multiple machines and real time supervision to observe the sequence and the types of log

events that were being recorded in response to each event in the IIT-BombayX courseware pages. A sample video was opened in a particular window, and it was played, closed, stopped, rewinded, fast-forwarded, closed abruptly and then reopened - and we even went as far as to remove the internet connections and make and break the connections to see how the events were being generated. The log files were also analysed by hand, as were the results of running specific queries on the UserSessionOldLog table. Based on these, we were able to understand about the basic and ideal anatomy of a session of a video playing, right from the start when the video is loaded, upto the point when the user either closes the page or goes to another video.

All the video events are signified by the eventType attribute value 'video'. For each of the video events, the set of events can be as follows:

- load_video : Which indicates the loading of a video on a particular page, or a particular lesson.
- play_video : Which indicates the action of a user having clicked on the video on the webpage to start playing it.
- pause_video : Which indicates the action of a user having clicked on the play/pause button to pause the video in the middle of playing it, without stopping it or ending the video completely.
- seek_video : Which indicates the action of the user in using the mouse to drag the scroll button of the video/ the video pointer forwards, backwards, or moving it back and forth throughout the video to start playing the video at a different point in the video.
- stop_video : Which indicates the fact event of the video playing all the way till the end point, and then stopping, such that playing it again would mean that the video would start from the beginning.
- save_user_state : The event which records the last point in the video till which the video was watched by the user. This event is fired in a lot of different cases, which will be discussed later.

Other than these events, we also considered an additional event called 'page_close', which comes under the eventType 'navigation'. This event is responsible for actually determining whether the user has is still on the page containing the video, or has actually switched to some other video and has left the earlier one. This event needed to be recorded, as it indicates the point of end of a user's activities for the time being, until the next time the video is loaded by the user for watching, and accordingly note the time watched by the user upto the closing of the page.

Let us now talk about the basic anatomy of a video playing event. The following diagram depicts a simple and very simplified flow of events for a single session spent by a user watching a particular lecture video 8.2.

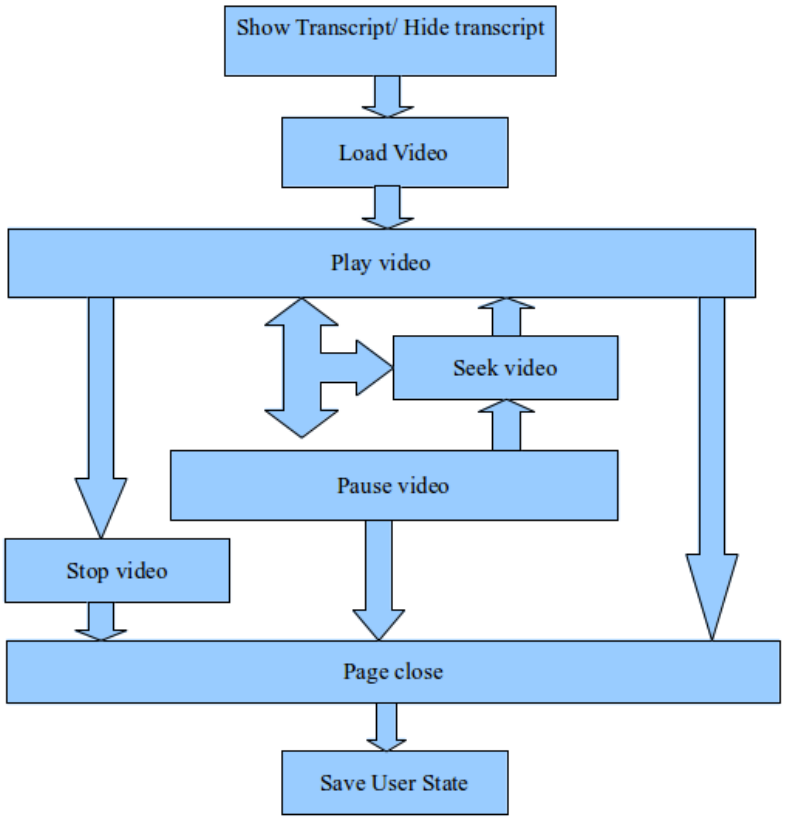


Figure 8.2: A simplified and ideal sequence of video watching.

As shown, the life of a video session begins when a user (generally a student) clicks on a link to go to a week or a lesson and lands up on a page which contains a video. As soon as the page is loaded, the video is loaded on to the browser side, and the several 'video' type events are triggered, such as: show_transcript, translation_transcript, hide_transcript, and of course, load_video. Usually, this event is followed by a save_user_state event, which records the last point of the video which was watched by this user. In case this is the first time this video is being watched, the currentVideoPosition attribute it would be set to 0.0; otherwise, it would correspond to the last position of the video upto which the user had watched and has been recorded by the last save_user_state event for the video. Once the video has been loaded, in the ideal case, the user would go on to play the video, and go on to execute the sequence of normal video processes as show in the diagram. He may play, and then pause; or he may play, seek, and continue playing; or pause, seek, and then start playing, and so on —— there are lots of such permutations of these events.

It should be noted at this stage that each pause video event would be followed by a save_user_state event, which records the last position upto which the video has been watched by the user, at the time of the pause. Also, when a seek_video event occurs, ideally, there is a save_user_state event which is fired which saves the last point upto which the video was

watched before it was the seek operation took the video scroll to a different point in the video for the user. Following this, the video may continue playing or remain paused, as it were before scrolling backwards or forwards. Finally, when the user has finished watching the video, the stop_video event is triggered, which records the fact that the video has ended. Following this, the user may start watching the video from the beginning again, or seek to a particular point in the video and start the same activities as before; or as in the normal, ideal case — go to another video's page, or a different lesson or week on the courseware. This would trigger a page_close event, which would be recorded by the event logs to give the page which has been closed by the user. Additionally, this would also trigger the event 'save_user_state' for the video in concern, to record the last location in the video which was playing, or which had been paused by the user/student. Thus, the simple anatomy of the video session is discussed.

However, as we discovered upon examining the logs for individual users arranged in chronological order by the MySQL queries used on the UserSessionOldLogs table — the sequence of events of the users were not as simple as the appeared to be. While the basic outline of the events stays the same, we have to account additionally for the fact that the users may skip to different parts at different times, may open new videos in new tabs on the browser —– or even certain errors and discrepancies in the events that were quite obviously, due to factors beyond the control of someone who is only looking at the logs generated. The errors included mistaken orderings of events, or events which were recorded at far different positions in the logs than where they were supposed to be. Most of these events would be due to the disturbances in the networks, so that successively generated events reached the record logs in an order different from the order generated. It could also be due to the scheduling behaviour of the tasks and the processes by the servers, which would again result in a similar situation. An example of this could be as follows: a save_user_state event, instead of appearing after the page_close event, appears before it in the log tables. Among other explanations for this discrepancy, one of the basic reasons why this could occur is because of the network errors. Thus, as part of the data model, we also had to develop logic to deal with such sequences of events appropriately, so that we could correctly extract the desired feature data we are desperately trying to get.

Basically, our plan has been to observe these video events and and identify the sequence of the events for a particular student to determine whether the student has been watching the video or not - and if so, then how long he has been watching, which parts he has been watching, and whether he has skipped to certain parts in the video, or even repeated watching a certain part of the lecture video. In other words, the video events were closely analysed to get an overall view of the student's ac-

tivities and interactions with the video in a single session, as discussed previously. For this we needed to track when the video was played and paused, and when the page was being loaded or the page was being closed by the user.

Now, we proceeded to design an algorithm to extract this data from the log files, examined in order of the createDateTime field in the tuples. We aimed to cover all the tuples in the result in a single traversal, and calculate the data for the different features for the video. In order to capture the different regions in the video that were visited by a user (and subsequently, all the users), we divided the entire duration of the video into a number of timeFrames, each of which was of width 4 seconds. So, the time frames were like: 0-4secs, 4-8secs, etc. The exact width of the timeFrames is still a matter of optimisation, and after running several examples and simulations, we can decide on the correct width of the timeFrames which would be most apt to capture the student activities. Each of these timeFrames were like a bucket, which were incremented each time they were traversed during a session of the video being watched by the user. The algorithm sequentially went through all the events, one by one, and maintained a track of the state of the video being watched internally through separate flag variables. Variables were also used to keep track of the range of the video that was watched by the user in a part of the video session, so that we could calculate exactly what part of the video was watched by the user until a change occurs in the behaviour of the user. A change in the behaviour of the user is explained thus: if a user, for example, plays a video, then pauses it, or seeks to a particular region of the video, or even closes the page —– then we would account it as a change in the behaviour of the user. It does not mean that the pattern of behaviour of the user changes, — it simply means that the user does something else other than simply watching the video, and this leads to a certain disruption in the normal viewing activity. hence, going in an incremental or rather, a step–by–step fashion, we will need to keep track of all the time in the video that the user has watched upto the time such a disruption occurs. each time such a disruption occurs, we may or may not change the internal state of the video (as maintained by our algorithm), and we increment the bucket for all the timeFrames in the video that have been covered upto the point of traversal. The following is a list of the steps and the disruptions that we considered while designing the algorithm, and how we dealt with the disruption to appropriately extract the time period in the video that has been watched by the user.

- **From the time a video is loaded (namely, encountering the 'load_video' event, we consider the video to be in the paused state. We simply initialise our internal pointers to the time ranges, and wait for the video to start playing.**
- **As soon as the video starts playing, we note the starting point of the video playing event and keep on watching the successive times**

(or timeFrames) in the video the user watches upto. This continues as long as the video is playing, and is indicated internally by setting flags to indicate so. This is because the IITBombayX system, as we said earlier, pulses the events such as play and pause several times second, and over times times too. this worked to our advantage, as it allowed us to explicitly track all incrementally the point upto which the user had watched the video at that point.

- **However, if it so happens that the same video is being watched simultaneously by the same user in separate tabs, then it would be necessary to capture all such events, and the same time Frames that are being covered by the user (because the effort counts!). The order wouldn't matter, as long as all the timeFrames are being effectively covered by our algorithm. So, as a special-case-handling mechanism, in case we noticed a different play_video time for the event, we considered a new video watching session was in order — so, we counted the time upto that point, and restarted a new video watching session from that point.**
- **In the most general case, the interval between a play and a pause event was considered as the time spent by a student. This is explained intuitively.**
- **Similar to the case as discussed with the play_video events, we would also need to track the pause_video events for videos playing in other tabs as well. In the general case, the student would quite obviously close the video he is watching in one tab before switching to the other one. However, he might even watch them simultaneously, for the purpose of comparing, and hence we would need to consider this effort being put in by the user. So, in this case, we wait and note for other different pause_video events that may have occurred, and accordingly, account for the time watched by the students.**
- **The time spent upto a seek was considered the time spent by a student, and the final point of seek marking a region that is probably being visited a second time (although, it may be a case of seek forward – where a student skips past a part of the lecture to go to a different part of the video, because he/she already has the knowledge being imparted in that part). In any case, we will consider the video, if 'playing', to go into a pause state immediately, until the next play_video event for the video. The time spent by the user upto the start of the seek_video event (in which case, the start point of seek would be given by the oldVideotime attribute o the log entry) is then recorded as the region of the video watched by the students. if the video is already paused (due to being paused prior to seeking by the user, or simply because a seek event has already happened for the video by the user), then we only account for the change in the starting time (or timeFrame) of the video — indicating the point from which the user would start playing the video, if there is a new play_video event.**
- **In case of a stop_video event, we simply account for the times in the**

video that have been watched by the user upto the end point of the video, and consider the video closed. of course, the save_user_state events would account for the current time upto which the upto which the video has been watched, thus keeping an additional track or check on the video activity. In the beginning of the playing of the video, this would indicate the point where the video was stopped earlier.

- **Finally, once all is said and done, we come to the situation where we the video and page(or tab) is closed for good. This may occur once the video has stopped playing, or is paused, or even while the video is playing. In all of these cases, the closing of the page of the video is recorded by the IITBombayX platform as being a page_close event, as mentioned before, and is immediately (usually) followed by a save_user_state event. As we have mentioned earlier in the section on error detection, we also had cases beyond our control where the events appeared out of the logical order of occurrences — in this case, save_user_state appearing before the page_close event. So, in the processing stage prior to the data model, we had reordered the save_user_state event that was most relevant related to the page_closed event following right after the page_close event.**
- **In case the page of the video is closed and the video is paused, we do not have anything to worry about. We have already updated the regions of the video that the student has watched up to the pause_video event. However, if the video is still playing when the page is closed, then we would do better by counting the time in the video upto the point where the page_was closed. This can be indicated by the fields in the save_user_state event associated with that page_close event.Of course, the internal flags would be set to indicate that indeed the page has been closed, and that the video is currently paused**
- **As a final correctional measure, we included the situation where the page_close event has not been recorded for the last video session of the user in the logs. However, we still need to include the time watched by the user — so, we consider the time watched by the user upto the last point in the video.**

From the timeFrameBuckets list, we can get the count for the accesses to different regions of the video. The process is thus continued for all the users for a particular video, and ultimately, for all the videos in a particular course, as is given by the logs provided. Thus, the output is produced by our program, which are basically sets of tuples for every video watched in the logs, every user who has watched it, and the timeFrames in the videos. At the prototype stage, the feature data generated after the data-model stage would be would be tuples or entries in a MySQL table, which would then be used for inference generation, or visualizations (in words, out final steps of work). The tuples would have have the following characteristics:

- **Primary key : ⟨ an incrementing unique numeric id ⟩, videoSysId, userName, timeFrameId**
- **Non-key attributes : frameAccessCount and frameDuration**

Once the table is populated with this data, as we said, we can generate our summaries from this table, to be used by visualizations and for learning methods, reports, and so on. Some of the possible summary information we can extract from this data could be as follows:

- **Total time spent by a user on a particular video : We could execute an aggregate query on the final summary data to extract the total amount of time that a user has spent on the total video, by summing over all the timeFrames and all corresponding frameDuration entries for that particular video in that course.**
- **Total time spent by all users in a particular timeFrame : An aggregate sum function could be executed on a particular timeFrameId, individually for all the users who have watched that particular video timeFrame. Typically, this would be a SUM function executed on the frameAccessDuration attribute for the selected timeFrameId for all the users for a video, grouping the entries by the videoName and userName attributes in the summary tables.**
- **Total number of accesses to a particular timeFrame : An aggregate sum function could be executed on a particular timeFrameId, individually for all the users who have watched that particular video timeFrame. Typically, this would be a SUM function executed on the frameAccessCount attribute for the selected timeFrameId for all the users for a video, grouping the entries by the videoName and userName attributes in the summary tables.**

As stated earlier, in order to develop the prototype of the data model and test run the algorithm, for correcting and perfecting it, we needed to run the program on some sample test, but very real data. Thus, we ran our program on the videos of the CS101.1x course. We first tested this on the top 10 most watched videos in the CS101.1x course, and on the top 50 viewers for the corresponding videos. The queries used to find the list of the videos and the users are similar to the ones we have used before for extracting the data for a single video and for a single user. The process was executed repeatedly for the different videos and the users to extract the final feature data for the videos.

The following graph 8.3 was plotted on the basis of the data obtained by running the module on a single video, and its top 50 users. The video in consideration had the videoSysName = '67a8559582864d6a8148e2ef5c997e8f', or otherwise known as the video on Elementary Graphics, in the CS101.1x course.

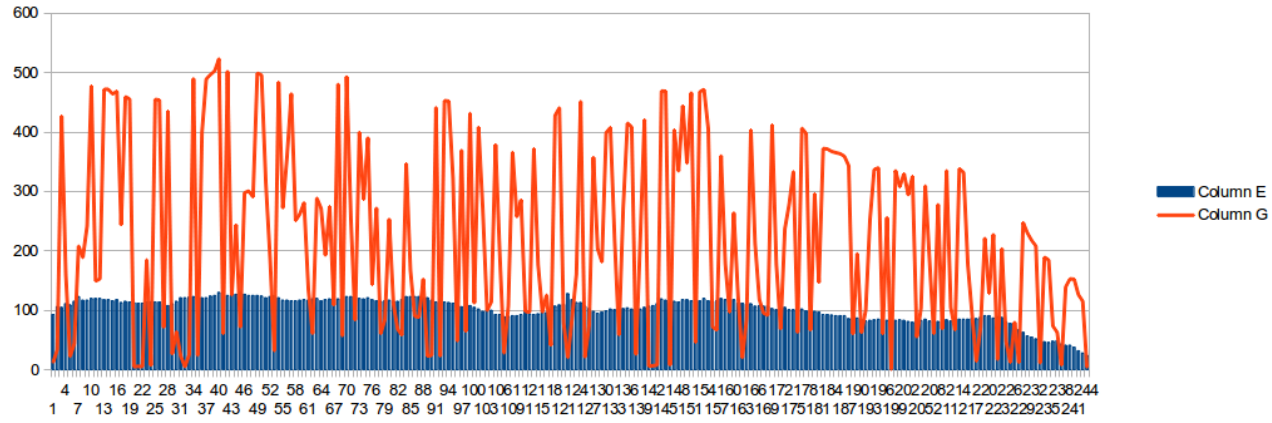


Figure 8.3: An example graph. Legend:X-axis - Time frame number. (dimensionless) Red curve - The total time spent by all the users in a particular timeFrame. (seconds) Blue Columns - The number of accesses to those timeFrames. (dimensionless) .

8.4.4 Final Implementation and visualization

Once the data model was finalized and found to be working properly at a prototype level, we took it the project to the final step — implement the data model in Spark to draw the data from populated Hive tables containing the cleaned Users' log data, put the processed data back into the Hive tables, and finally process the result data to create summary data that would be populated into the MySQL Analytics database tables, which would be used for visualization on the IITBombayX platform dashboard for the instructors.

The final data module would be working as a live application in a Big Data environment, as has been said before — since it will deal with tens of gigabytes of log data generated by the IITBombayX platform. To appropriately handle this, we would require a tool that can efficiently work with such vast amounts of data, but do it efficiently nevertheless. Spark does that for us, using the concept of RDDs, and allows us to run queries similar to SQL on the RDDs it creates from the data in the memory. To work with such a setup, an installation and proper configuration of Hadoop (for distributed filesystems), Hive (for creating data warehouses), and Spark (for running SparkSQL queries and executing operations on the data). Once this is done, the workflow of the operations, and in a more broader sense, the algorithm for the Spark programs were decided upon.

8.4.4.1 Extracting the data from Hive tables

The data that we originally worked upon (the UserSessionOldLog table) was a MySQL table. However, working in a Big Data environment

requires the maintenance and access of a fast data warehouse, such as Hive. In order to transfer the data from the MySQL database to the Hive tables for the same, we used the Sqoop utility to transfer the data. The command for the Sqoop utility can be writen as follows:

```
sqoop import --connect jdbc:mysql://localhost:3306/IITBxDataAnalytics
             --username root --password '' --table UserSessionOldLog
             -m 1 --hive-import --hive-table default.userlog
             --warehouse-dir /user/hive/warehouse
```

The command can be explained as follows. We specify the connector and the database to connect to using the –connect option, and together with it, the username and password to access the data base, and the table to be imported. Then, we specify the name of the destination table in the Hive database and where the warehouse for the hive tables is located on the Hadoop HDFS (this can be verified by going to the Hadoop administration page for the namenode, and then browsing the filesystem. The process was completed successfully for us, when we also transferred the CourseVideos table and a sample of the UserSessionOldLog table to the Hive databases.

While it is known that Spark runs very fast, it is required that the data must be in the main memory in order for the processing to be very efficient. So, this required us to specify a single query to draw the user data from the Hive tables in a single go, and perform further processing on it 'in-memory', using the concept of RDDs where useful, and then collecting the data from the RDDs when they needed to be used directly. So, work was done to optimize and perfect the right queries that would fetch the necessary event data in a single go from the databases. The query which finally worked was as follows:

```
SELECT DISTINCT orgname, coursename, username, modulesysname,
                eventtype, eventname, oldvideotime, currvideotime,
                createdatetime
FROM userlogsmall
WHERE eventtype='video' OR (eventtype='navigation' AND
                            eventname='page_close')
ORDER BY orgname, coursename, username, createdatetime
```

This query can be explained thus. We divided all the events based on the organisation which conducted the course, the name of the course, the user who was involved in the event, the module being used or involved, and the eventType and eventName for the corresponding event, together with the oldVideoTime and currVideoTime if the events are of the type 'video', as is required by us. The events are then ordered by the organisation, coursename, the username, and then the time of the event — so that while processing the events, we can not only determine the

temporal sequence of the events, but they will be processed depending upon the user who was involved in the event, so that all the behaviour of the user can be taken into account while processing his behaviour during the watching of a single video.

To begin with, in order to test the final implementation of the data model, we selected a subset of the original records to contain only the records of the top 30 most active users on the platform, using the following MySQL/Hive query (the records were first isolated into a separate table 'userlogsmall' in MySQL, and then exported to Hive via Sqoop):

```
create table userlogsmall (
    select USOL.* from UserSessionOldLog USOL,
    (select userName, count(eventType) cet
        from UserSessionOldLog
        where userName is not NULL and userName not in ('')
              and courseName='CS101.1x' group by userName
        order by cet desc limit 30) t1
    where USOL.userName = t1.userName and courseName='CS101.1x')
```

8.4.4.2 Filtering of events per user and per video

Following this step, the events for each user and each video were separated out from the event logs in the order in which they have been extracted. The idea here was that instead of extracting the events per video and then per user, we extract them per user, and then per video because that captures any changes of patterns in the user's behaviour in watching the videos. The result of this process was a single key for each orgname, coursename, videoname and username (for every user watching a video in a course), and its corresponding value being a list of events for that user and that video in that course. A number of conditions were considered while computing these lists by sequentially going through the sequence of events extracted. They are outlined as follows:

- **For each log event of the type 'video', we are sure of the video id and the user who has watched it. So, it is immediately appended to the list of events for the corresponding key (i.e., (orgname, coursename, videoname, username)).**
- **If a log event is the very first log event in the sequence of events for a particular user (which is also the case of the first log event in the result set of the queries executed before), then we check if the event is a not a save_user_state or a page_close event. If so, then it is a video event with a definite username and videoname, we add it to the list of events for that key constructed. Otherwise, we ignore it, as we have no use for an event taken out of its context. For example, a page_close event maybe the first event in the result set for a particular user, but it may be the page_close event for a**

module other than a video —— and this is the case, as has been observed.

- **If, however, we encounter a save_user_state event or a page_close event for a user we are currently working with, we keep track of the last (user,video) key we were working with, and add the event to the list corresponding to that key, because it is most likely related to that (user,video) combination.**

The above operation was carried out all in Python code written in the Spark module. This is because, performing join operations, given the volume of records we are dealing with, will result in a huge amount of memory usage and processor time, which can easily be tackled if we instead go for the sequential verification strategy selected over here. The events are verified, and associated with the corresponding video and the user watching the video.

It should be noted at this stage that in the table for the course videos, there were certain errors int e details of the videos recorded. In particular, the video with the id '2017625218ba4309b4cd42309f5d82e2' had neither youtube id nor a title name. Hence this video had no duration calculated from it, and the events corresponding to this video could not be considered. In order to deal with this, we first allowed our filtering algorithm detect and accumulate the events for the video and for all the users that have watched the video. Then, the events for the corresponding keys having that video name/id were eliminated from the final list of key-value pairs, the value here being the list of events for the user and the video.

8.4.4.3 Processing and generation of results for final summary table

Now that the events have been filtered out and associated with the corresponding videos and the users, it is time to process the events and generate the statistics for the timeFrames that have been visited and accessed by the users while watching that video. At this stage and in the summary stage, the concept of MapReduce workflow was used to speed up the process of analysing the data generation and processing tasks. The concept of MapReduce revolves around the fact that the data can be converted into a set of key-value pairs by passing it to a mapper function, and these key value pairs are then 'merged' together using a reducer function to create the final result. It is a workflow methodology which basically uses parallelism, and given the scale of data we are working with, it worked to a great advantage for us.

Prior to this, the RDD (Resilient Distributed Database) was created for each of the (key,value) pairs, where the key was the a combination of the orgName, courseName, videoSysId and the userName, and the

corresponding value was the list of events for that key. The key value pairs would look something like this:

```
(
     (u'IITBombayX', u'CS101.1x', u'fa1f6040f46a43298cc25fc33db89a83',
     u'Shrikrishna'),
     [
       (u'video', u'load_video', 0.0, 0.0, u'2015-02-17 08:22:03.0'),
       (u'video', u'play_video', 0.0, 0.0, u'2015-02-17 08:29:32.0'),
   (u'video', u'pause_video', 0.0, 11.525, u'2015-02-17 08:29:32.0'),
      ...... and so on....
     ]
)
```

The RDD parallelized this combination to create the key,value pairs, which were then sent to a mapper function called processing(), which essentially performs the analysis required on the key-value pairs, as described in the section on the development of the data model prototype. The single function contains all the code needed to generate the details for the timeFrame access for a single video and a single user who has watched it. The code for these parts are given as follows:

```
eventMap=sc.parallelize(eventDict.items()).map(lambda p: (p[0], p[1]))
videoDetails = eventMap.map(processing)
```

While the input consisted of key value pairs where the value was a list of events, the output produced key-value pairs having the same keys, but having the timeFrameBuckets list as the value. To recall, the timeFrameBuckets list is a list of bi-tuples (x, y), where x denotes the number of accesses to a particular timeFrame, and y denotes the time spent by the user in that timeFrame of the video. After processing, the data would look something like this:

```
(
   (u'IITBombayX', u'CS101.1x',
    u'fa1f6040f46a43298cc25fc33db89a83', u'Shrikrishna'), [[4, 16.0],
                                                          [4, 15.05],
                                                          [2, 8.0],
                                                .....and so on ....]
)
```

This step thus produces the necessary data of the analysis, which can can be used various kinds of purposes later on, such as visualizations, summary metrics, and learning activities. Also, this data could be stored as itself, in the form of a tuple in the Hive tables, so that each time a new visualization needs to be created, the processed (but not summary) data can be accessed from the Hive tables.

In case of final implementation of the data model and processing module, the processed data generated would be stored in the corresponding Hive table as tuples having the following fields:

- **Primary key : ⟨ an incrementing unique numeric id ⟩, orgName, courseName, videoSysId, userName, timeFrameId**
- **Non-key attributes : frameAccessCount and frameDuration**

One of the possible visualizations that can be generated from this data is the total number of times accessed and the total time spent by users in a particular timeFrame of a particular video. What this would give is an overall analysis of the number of views and the time of the views on the video in question. An instructor would be able to use this analytical data to determine what are the regions of the videos that are being watched by users (who have watched this video), most frequently. Accordingly, these could be the regions of investigation for the instructors, as these are the regions of difficulty for the students as stated in our hypotheses.

The summary data is generated as follows. First, the results of the previous analysis stage would be converted into an RDD through the parallelise function, and then converted into appropriate key-value pairs —- only this time, the key would not contain the user id. Instead, it would contain the videoFrame number, because we are required to do a sum over all of the users for a single videoFrame in a single video. This is done by converting the key and list combination as given before into a list of key-value pairs, and then reducing all of the lists into a single list using a reduce() job, which basically concatenates the lists together.

Following this, the result of reduce() was again parallelized to get an RDD, on which, we ran a final map-reduce job on the keys as stated before, but this time, the reducer function we used added the tuples together to sum up the total number of accesses and the total time spent in a timeFrame. The results of this map-reduce were then written to the MySQL summary table video_difficulty_analytics using Python code.

The workflow stated can be summed up in the two code lines as follows, which illustrates the power of Spark and of map-reduce:

```
videoUserTimeFrameAggregate
      = sc.parallelize(
              videoDetails.map(
                   lambda x:
                   [ (x[0], i, x[1][i]) for i in range(0,len(x[1])) ]
                      ).reduce(lambda a,b: a+b)
              )

#x[0]--> key as used in processing(),
```

```
#i--> timeFrame id, x[1]-->timeFrameBuckets

result = videoUserTimeFrameAggregate
                .map( lambda x:
                ( (x[0][0], x[0][1], x[0][2], x[1] ), x[2] ) )
                .reduceByKey(lambda a,b: (a[0]+b[0], a[1]+b[1] ) )

#x[0][...]: orgName, courseName, videoSysId,
#x[1]--> timeFrameId, x[2]-->tuple of count and timeSpent.
```

The summary data generated for creating the visualizations would be stored in the corresponding MySQL table (video_difficulty_details) as tuples having the following fields:

- **Primary key : id ⟨ an incrementing unique numeric id ⟩, orgName, courseName, videoSysId, videoFrame**
- **Non-key attributes : count and timeSpent**

The summary data was then drawn from these two tables using an R data visualization script using the googleVis library to plot a graph on the analytics dashboard. The graph had two y-axes: one for the timeSpent on the video timeFrames (indicated on the X-axis), and one for the 'count' field, giving the number of accesses to the timeFrame in the video by all the users.

8.5 Future Work

Although we have so far been able to visualize the feature data such as time spent in different regions of the video, and the number of times people have visited the different regions of the video. However, the project has greater potential than just this. Once we have the data to work with, we determine the patterns in the overall viewing activity of not only a single video, but also all the videos in a particular course undertaken by an instructor. One possible approach could be to first determine clusters of high video activity (namely, large number of views and number of accesses) across the various timeFrames of the videos. Once these clusters have been determined, one could possibly create a type of scheme for determining the score of such a cluster. The reason for this being that we will have clusters in all the videos, depicting the regions of difficulty in these videos. So, we need to be able to compare these clusters with each other, in order to determine which clusters indicate a higher level of difficulty than the others. The clustered regions for different videos in a course could be compared with each other in order to find the regions of diffculty, and report them to the instructor. The instructor, as said in the beginning, would receive notifications for the same, and take appropriate actions to alleviate the problem.

A further improvement to this work could be the automation of the recommendation and alert process. In such a system, the system itself would recognize the regions of difficulty in the videos using the method described above, and automatically recommend materials and course videos to the students, thus automating the educational platform to make it even more personalized for the people accessing the course, requiring inly the instructor to manage the course, monitor student activity, and address any queries the students may have regarding the lectures.

8.6 Technologies Used

A variety of technologies have been used by us during the development of this data model and final application module. In particular, we used the following during the mentioned stages of the development process:

1. **Prototype stage :**
   - **MySQL : The database management system used for storing the UserSessionOldLog and CourseVideos table, and in the final implementation, for storing the summary data extracted from the video events of the users.**
   - **Python : The language base used for programming in this project, and for prototyping and testing the data model.**
2. **Final implementation stage :**
   - **Hive : The data warehouse infrastructure utilising Hadoop as well as allowing access to files as RDDs for executing queries.**
   - **SparkSQL : The fast data processing engine, using in-memory primitives, and faster than Hadoop.**

Chapter 9

Integration

The final step is to integrate everything such that both data loading and data visualization can be done at a single place from web framework which requires no other software installation in clients machine.

To accomplish this, we have used Django - a python based web framework. Django can easily execute python queries, make a Django web template(interface) and has support of running Hive, R queries from it.It is also capable of queuing the processes i.e., if a process takes large time to execute, other processes are queued up and completed when the previous one has finished its task.

9.0.1 Django- Basic Design and Layout

9.0.1.1 Django project layout

mysite/

- **manage.py**
- **mysite/**
  1. **settings.py**
  2. **urls.py**
  3. **wsgi.py**

These files are as follows:

- **mysite/: The outer mysite/ directory is just a container for your project. Its name doesnt matter to Django; you can re- name it to anything you like.**
- **manage.py: A command-line utility that lets you interact with this Django project in various ways. Type python man- age.py help to get a feel for what it can do. You should never have to edit this file; its created in this directory purely for convenience.**
- **mysite/mysite: The inner mysite/ directory is the actual Python package for your project. Its name is the Python pack- age name youll need to use to import anything inside it (e.g. import mysite.settings).**

- **A file required for Python to treat the mysite directory as a package (i.e., a group of Python modules). Its an empty file, and generally you wont add anything to it.**
- **settings.py: Settings/configuration for this Django project. Take a look at it to get an idea of the types of settings available, along with their default values.**
- **urls.py: The URLs for this Django project. Think of this as the table of contents of your Django-powered site.**
- **wsgi.py: An entry-point for WSGI-compatible webservers to serve your project. See How to deploy with WSGI (https://docs.djangoproject.com/en for more details.**

9.0.1.2 Libraries imported

- **To return a HTTP object**

```
from django.http import HttpResponse
from django.http import HttpResponseRedirect
```

- **For a template**

```
from django.template import Template, Context
from django.template import *
from django.shortcuts import render
from django.template.loader import get template
from django.shortcuts import render to response
from django.templatetags import *
from django.template.base import Library
```

- **For using any date-time type object**

```
import datetime
```

- **For using csv**

```
import csv
```

- **For using POST as the method to send data via form**

```
from django.core.context processors import csrf
```

- **For including other processes in a process**

```
import subprocess
```

- **To include other views of the project**

```
from mysite import *
```

- **For including other settings**

```
from django.conf import settings
```

- **For zipping up different lists together so that they can be accessed together**

```
from itertools import izip
```

9.0.1.3 Folders to be manually added

- **static: Inside mysite/mysite/ to store external static files like javascript libraries, .csv files, images etc**
- **templates: Inside mysite/mysite/ to store html templates**

9.0.1.4 New settings.py file

- **Addition of path to the static folder**

```
STATIC URL = âĂŹ/static/âĂŹ
STATICFILES DIRS = (âĂŹpath to directory/mysite/mysite/static/âĂŹ,)
```

- **Adding csrf library so as to enable data transport using âĂİpostâĂİ**

```
MIDDLEWARE CLASSES = (
âĂŹdjango.middleware.csrf.CsrfViewMiddlewareâĂŹ,
other classes
```

Above class should be above any other class that uses csrf

- **Addition of path to templates folder**

```
Add TEMPLATE DIRS =
(âĂŹpath to folder/mysite/mysite/templatesâĂŹ,)
```

9.0.1.5 Points to remember while using Django

- **Django folder should have read and write permission by all users which can access it on the same system**
- **To make django run on server, start django server by typing**

```
python manage.py runserver 0.0.0.0:8000
```

- **For accessing any external static file Add**

```
 {% load staticfiles %}
```

once in the top of the template which uses static files. âĂİA.csvâĂİ will be written as

```
âĂİ{% static âĂŹA.csvâĂŹ %}âĂİ
```

- **To access python variables inside template, use**

```
{% variable name %}
```

- **To render any template, use**

```
return render(request, âĂŹtemplatename.htmlâĂŹ, {âĂŹvariable nameâĂŹ:
"variable value",
âĂŹvariable2âĂŹ: variables value})
```

variable name implies the variable which will be used inside the templates and its value to be substituted is passes while rendering the template

9.0.1.6 Django views syntax

```
import library1
import library2
def function1 name:
- description
def function2 name:
- description
..
```

9.0.2 urls.py

- **Contains the url of various views and their respective templates**
- `urlpatterns = patterns(âĂŹâĂŹ,`
  `other urls`
  `url(râĂŹ^hello/$âĂŹ, "mysite.views.function1"),`
- **denotes start of the url and denotes the end. Therefore,above example means that if url contains only âĂİhelloâĂİ, execute function1 of views view.**

Table 1.1: Educational Background chart

Label	Student Response	Description
None	None	No Formal Education
Primary	Elementary/primary school	Initial schooling lasting approximately six years
Middle	Junior secondary/junior high/middle school	Continuing basic education lasting two to three years
Secondary	Secondary/high school	More specialized preparation for continuing education or employment lasting three to four years
Associate	Associate degree	Completion of two years of post-secondary education
Bachelor's	Bachelor's degree	Completion of four years of post-secondary education
Master's	Master's or professional degree	Certification for advanced academic or occupationally specific education
Doctorate	Doctorate	Advanced qualification for original research

Figure 9.1: Special syntaxes to be used for specifying the url format

9.1 Visualization Using Google Charts:

Google Charts provides a perfect way to visualize data on your website. From simple line charts to complex hierarchical tree maps, the chart gallery provides a large number of ready-to-use chart types. The most common way to use Google Charts is with simple JavaScript that you embed in your web page. You load some Google Chart libraries, list the data to be charted, select options to customize your chart, and finally create a chart object with an id that you choose. Then, later in the web page, you create a ÂądivÂ£ with that id to display the Google Chart.ThatâĂŹs all you need to get started. Charts are exposed as

JavaScript classes, and Google Charts pro- vides many chart types for you to use. The default appearance will usually be all you need, and you can always customize a chart to fit the look and feel of your website. Charts are highly interac- tive and expose events that let you connect them to create com- plex dashboards or other experiences integrated with your web- page. Charts are rendered using HTML5/SVG technology to pro- vide cross-browser compatibility (including VML for older IE ver- sions) and cross platform portability to iPhones, iPads and Android. Your users will never have to mess with plugins or any software. If they have a web browser, they can see your charts. All chart types are populated with data using the DataTable class, making it easy to switch between chart types as you experiment to find the ideal appearance. The DataTable provides methods for sorting, modifying, and filtering data, and can be populated directly from your web page, a database, or any data provider supporting the Chart Tools Datasource protocol. (That protocol includes a SQL-like query language and is implemented by Google Spread- sheets, Google Fusion Tables, and third party data providers such as SalesForce. You can even implement the protocol on your own website and become a data provider for other services.)

The various gvis functions read a data.frame and create text output referring to the Google Visualisation API, which can be included into a web page, or as a stand-alone page. The actual chart is rendered by the web browser using SVG or VML.
The following are the charts used for visualization:

1. **Line Chart**
   Usage:

   ```
   gvisLineChart(data, xvar = "", yvar = "", options = list(), chartid)
   ```

2. **Column Chart**
   Usage:

   ```
   gvisColumnChart(data, xvar = "", yvar = "", options = list(), chartid)
   ```

3. **Pie Chart**
   Usage:

   ```
   gvisPieChart(data, labelvar = "", numvar = "", options = list(), chartid)
   ```

4. **Bubble Chart**
   A bubble chart is used to visualize a data set with 2 to 4 dimensions. The first two dimensions are visualized as coordinates, the 3rd as color and the 4th as size.

   The bubble chart is rendered within the browser using SVG or VML and displays tips when hovering over points.
   Usage:

   ```
   gvisBubbleChart(data, idvar = "", xvar = "", yvar = "",
   ```

5. **Motion Chart**
   Usage:

```
gvisMotionChart(data, idvar="", timevar="")
```

6. **Stacked Area Chart**
   Usage:

```
gvisSteppedAreaChart(data, xvar = "", yvar = "", options = list(), chartid
```

7. **Geomap**
   A geo map is a map of a country, continent, or region map, with colours and values assigned to specific regions. Values are displayed as a colour scale, and you can specify optional hover-text for regions. The map is rendered in the browser. Note that the map is not scrollable or drag-gable, but can be configured to allow zooming.
   Usage:

```
gvisGeoMap(data, locationvar=âĂİ, numvar
```

8. **Table**
   The gvisTable function reads a data.frame and creates text output referring to the Google Visualisation API, which can be included into a web page, or as a stand-alone page. The actual chart is rendered by the web browser. A table that can be sorted and paged. Table cells can be formatted using format strings, or by directly inserting HTML as cell values. Numeric values are right-aligned; boolean values are displayed as check marks. Users can select single rows either with the keyboard or the mouse. Users can sort rows by clicking on column headers. The header row remains fixed as the user scrolls. The table fires a number of events corresponding to user interaction.
   Usage:

```
gvisTable(data, options = list(), chartid, formats = NULL)
```

9. **Add edit button for on the fly customisation:**
   Usage:

```
gvisLineChart(df, "country", c("val1","val2"),

options=list(gvis.editor="Edit me!"))
```

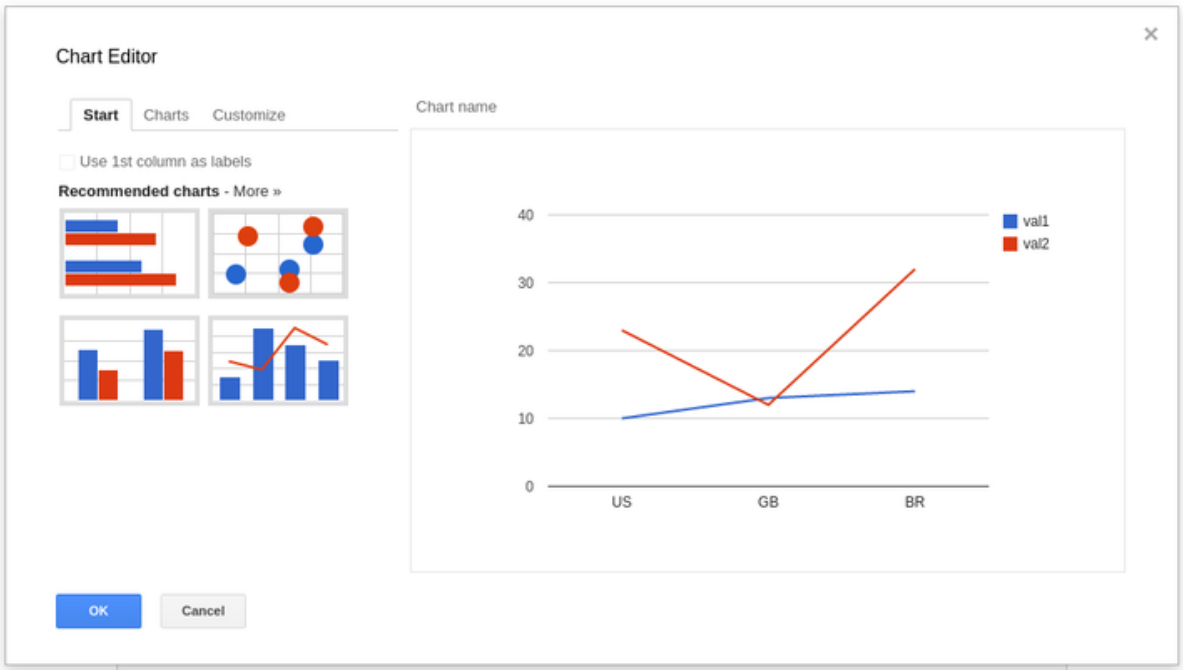


Figure 9.2: Graph with edit me option

Chapter 10

Bottleneck

The OpenEdX-Analytics-Pipeline was successfully installed and a hitherto unachieved aim of fetching data into the data api was achieved. The hindrance at this point was the inability to provide LMS authorisation to the available modules to get reflected on the dashboard.We tried searching for the settings of the IIT BombayX consisting of the authorization section or not.Also,we tried to make changes on the settings.py file of the LMS but eventually failed to get it reflected.This was one of the problem which we faced and could not resolved it.

We were successful in reaching the authentication page of LMS and passing the session back to the analytics system but there was a problem of ssl certificate authentication and the state value.

Chapter 11

Conclusion and Future Work

Data Analytics enables organizations to analyze a mix of structured,semi-structured and unstructured data in search of valuable bussiness information and insights.We have designed a system capable of uploading both the real time event log data(user interaction data) and summary data of IIT BombayX. The system is presently capable of analyzing the student interaction and produce various models like course-enrollment, course-activity (user engagement with the resources) , course performance and video analysis.

Our system would help the stakeholders (students as well as instructors) to analyse the data easily and derive useful information from it so that weak students could be helped to imporve and good ones can be praised. The future work in this area would be :

1. **Checking incremental update of the log data and Analytics Database table.**
2. **Making Operetaor Dashboard so that ETL and Model Population tasks can be automated.**
3. **Selection of log files can be automated.**
4. **Writing luigi task for Answer-Distribution and User-Navigation.**
5. **To come up with some more robust models like Recommendation and Prediction based on ML Techniques.**
6. **Working on the OpenEdX-Analytics-Pipeline to take from data-api-client to the dashboard.**