**Demo 1] Demonstration - Understanding MapReduce**

1] Download dataset in STAGING\_AREA

2] Pull it in LABS\_HOME/demos

3] Wait for the instructions from an Instructor

Dataset - demos/constitution.txt

Ref - HDPDev-PigHive-LabGuide-Rev 6.doc

**Lab 2] Writing down WordCount Program**

1] Download the rar in STAGING\_AREA location

2] Extract it

3] Pull the extracted folder in LABS\_HOME/wordcount

4] Wait for the instructions from an Instructor

Dataset - Lab2/wordcount.rar

**Lab 3] Refactor above WordCount Program to Apache Developer style**

1] Refactor your existing WordCount Program/application/project to apache Hadoop MR Developer style by referring the source code of WordCount from Apache

2] Create a new Eclipse Java project wordcount\_refactor

3] Keep only a single .java file that is having a main method, in your case it will be your WordCountJob.java

4] Put Your existing Mapper and Reducer classes as static inner classes, in above java file

5] Create a jar having name as wordcount\_refactor.jar

6] Use the same yarn jar command you used for your previous WordCount application

7] Confirm that it is working

Prerequisite- Lab 2 was implemented previously.

TimeLine - 45 Mins

**4] Distributed GREP**

Here attached the rar file. Perform following activities

1) Download the rar file in **STAGING\_AREA**, extract it

2) Create a folder grep in **LABS\_AREA**. Import the java files from step 1 to this folder

3) Create a project **grep** in Eclipse

4) Import the java file from step 2 in the same

5) Resolve the compilation errors by adding hadoop client side libraries in class path of the project

6) Implement all **TODO**s in the code

7) Look at the comments at the end of the file and create a jar and execute it

8) During implementation of the code refer the API documentation.

9) What differences you are finding in this code and previous WordCount code?

Dataset - Lab4/grep.rar

**Time Line =** 40 Mins

**5] Inverted Index**

Here attached the rar file. Perform following activities

1) Download the rar file in STAGING\_AREA location

2) Create a folder invertedindex in LABS\_AREA.

3) Create a project invertedindex in Eclipse

4) Import the java file in the same

5) Resolve the compilation errors by adding hadoop client side libraries in class path of the project

6) Implement all TODOs in the code

7) Look at the comments at the end of the file and create a jar and execute it

8) During implementation of the code refer to the API documentation.

9) Have you seen such types of applications?

10) Automate the running of this application by encapsulating it into a shell script/bash alias

Dataset - Lab5/invertedindex.rar

**Time Line = 40 Mins**

**Lab6] Inverted Index (Automation)**

1. Automate the entire process of building and running the application
2. Building - A process of compiling the \*.java files, creating \*.class files, incorporating it into \*.jar file
3. Refer the shell script in an application listed under Ref section

Dataset:- Lab6/MR\_Workflow/

TimeLine - Wed May 31, Lunch time

**Lab7] Implementing Counter**

1] Download an attachment in STAGING\_AREA location

2] Extract it

3] Pull it to LABS\_AREA/counter

4] Study the code

5] Create an eclipse project Counter

6] Import the java files from project folder in STAGING\_AREA in to this project

7] Create counties folder in your home on HDFS

8] Put all counties\_\*.csv files from local linux file system to counties folder on HDFS

9] Run the application with command as yarn jar average.jar <Main Class>

10] See the output of the job on terminal window and confirm whether counters you created are working or not

Study the code

**Time Line** = 30 Mins

Dataset - Lab7/Average.rar

**Lab8] Compression**

1] Download the rar file in STAGING\_AREA

2] Extract it

3] Pull the extracted folder in LABS\_AREA

4] Create a logfiles/ folder on HDFS and put the log files from project folder into it

5] Create an eclipse project compression

6] Import the java files from project folder in STAGING\_AREA in to this project

7] Remove the compilation errors by adding hadoop client side libraries

8] Create a jar compression.jar

9] Have a first Run using yarn jar compression.jar INFO command

10] Wait for the instructions from the instructor

**Time Line** = 30 Mins

Dataset - Lab8/Compression.rar

**Lab9] Streaming Job**

1] Download RAR in STAGING\_AREA

2] Extract it

3] Pull the extracted folder in LABS\_AREA

4] Wait for the instructions from an Instructor

Dataset - Lab9/MRStreaming.rar