Week-4	Pig commands – Pig Latin - Relational Operators: Loading, Storing, Diagnostic Operators (Dump, Describe, Illustrate & Explain)

4a For the given Student dataset and Employee dataset, perform Relational operations like Loading, Storing, Diagnostic Operations (Dump, Describe, Illustrate & Explain) in Hadoop Pig framework using Cloudera

Student ID	First Name	Age	City	CGPA
001	Jagruthi	21	Hyderabad	9.1
002	Praneeth	22	Chennai	8.6
003	Sujith	22	Mumbai	7.8
004	Sreeja	21	Bengaluru	9.2
005	Mahesh	24	Hyderabad	8.8
006	Rohit	22	Chennai	7.8
007	Sindhu	23	Mumbai	8.3

Employee ID	Name	Age	City
001	Angelina	22	LosAngeles
002	Jackie	23	Beijing
003	Deepika	22	Mumbai
004	Pawan	24	Hyderabad
005	Rajani	21	Chennai
006	Amitabh	22	Mumbai

Step-1: **Create a Directory** in HDFS with the name **pigdir** in the required path using **mkdir**:

$ hdfs dfs -mkdir /bdalab/pigdir

Step-2: The input file of Pig contains each tuple/record in individual lines with the entities separated by a delimiter (“,”).

In the local file system, **create an input file student_data.txt** containing data as shown below.	In the local file system, **create an input file employee_data.txt** containing data as shown below.
001,Jagruthi,21,Hyderabad,9.1 002,Praneeth,22,Chennai,8.6 003,Sujith,22,Mumbai,7.8 004,Sreeja,21,Bengaluru,9.2 005,Mahesh,24,Hyderabad,8.8 006,Rohit,22,Chennai,7.8 007,Sindhu,23,Mumbai,8.3	001,Angelina,22,LosAngeles 002,Jackie,23,Beijing 003,Deepika,22,Mumbai 004,Pawan,24,Hyderabad 005,Rajani,21,Chennai 006,Amitabh,22,Mumbai

Step-3: **Move the file** from the local file system to HDFS using **put (Or) copyFromLocal** command and verify using -cat command

$ hdfs dfs -put /home/cloudera/pigdir/student_data /bdalab/pigdir/

$ hdfs dfs -cat / bdalab/pigdir/student_data

$ hdfs dfs -put /home/cloudera/pigdir/employee_data /bdalab/pigdir/

$ hdfs dfs -cat /bdalab/pigdir/employee_data

Step-4: **Apply Relational Operator – LOAD to load the data** from the file student_data.txt into Pig by executing the following Pig Latin statement in the **Grunt shell**. Relational Operators are **NOT case sensitive.**

$ pig => will direct to **grunt> shell**

grunt> student = LOAD ' /bdalab/pigdir/student_data.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, city:chararray, cgpa:double);

grunt> employee = LOAD ' /bdalab/pigdir/employee_data.txt' USING PigStorage(',') as (id:int, name:chararray, age:int, city:chararray);

Step-5: **Apply Relational Operator – STORE** to **Store the relation** in the HDFS directory "/pig_output/" as shown below.

grunt> STORE student INTO ' /bdalab/pigdir/pig_output/ ' USING PigStorage (',');

grunt> STORE employee INTO ' /bdalab/pigdir/pig_output/ ' USING PigStorage (',');

Step-6: **Verify the stored data** as shown below

$ hdfs dfs -ls /bdalab/pigdir/pig_output/

$ hdfs dfs -cat /bdalab/pigdir/pig_output/**part-m-00000**

Step-7: **Apply Relational Operator – Diagnostic Operator – DUMP to Print the contents of the relation**.

grunt> Dump student

grunt> Dump employee

Step-8: **Apply Relational Operator – Diagnostic Operator – DESCIBE to View the schema of a relation**.

grunt> Describe student

grunt> Describe employee

Step-9: **Apply Relational Operator – Diagnostic Operator – EXPLAIN to Display the logical, physical, and MapReduce execution plans** of a relation using **Explain** operator

grunt> Explain student

grunt> Explain employee

Step-9: **Apply Relational Operator – Diagnostic Operator – ILLUSTRATE to give the step-by-step execution of a sequence of statements**

grunt> Illustrate student

grunt> Illustrate employee

4b	For the given Student dataset and Employee dataset, perform Relational operator – GROUP operations in Hadoop Pig framework using Cloudera

The **GROUP** operator is used to group the data in one or more relations. It collects the data having the same key.

grunt> Group_data = GROUP Relation_name BY Key;

Step-1: **Group the records/tuples** in the relation by age using **GROUP** command and verify.

grunt> group_std = GROUP student BY age;

grunt> Dump group_std;

grunt> group_emp = GROUP employee BY city;

grunt> Dump group_emp;

Step-2: View **Schema of the table after grouping** the data using the describe command as shown below.

grunt> Describe group_std;

group_std: {group: int,student: {(id:int, name:chararray, age:int, city:chararray, cgpa:float)}}

grunt> Describe group_emp;

group_emp: {group: int,employee: {(id: int,name: chararray,age:int,city: chararray)}}

Step-3: **Group by multiple columns** of the relation by age and city and verify the content.

grunt> groupmultiple_std = GROUP student BY (age, city);

grunt> Dump groupmultiple_std

grunt> groupmultiple_emp = GROUP employee BY (age, city);

grunt> Dump groupmultiple_emp

Step-4: **Group by All columns** of the relation and verify the content.

grunt> groupall_std = GROUP student All;

grunt> Dump groupall_std

grunt> groupall_emp = GROUP employee All;

grunt> Dump groupall_emp

Step-5: **Combinedly Group the records/tuples** of the relations student_data and employee_data with the key age and then verify the result.

grunt> cogroup_stdemp = COGROUP student_data by age, employee_data by age;

grunt> Dump cogroup_stdemp