	Write pig latin scripts to perform set and sort operation

Set Operation: UNION
UNION operator of Pig Latin is used to merge the content of two relations.
To perform UNION operation on two relations, their columns and domains must be identical.
Syntax: grunt> relationname3 = UNION relationname1, relationname2;
student1 = LOAD 'student1_data.txt' using PigStorage(',') as (studentid:int, studentname:chararray,percentage:int)
student2 = LOAD 'student2_data.txt' using PigStorage(',') as (studentid:int, studentname:chararray,percentage:int)
grunt> student = UNION student1, student2;
grunt> DUMP student

Set Operation: Join	
Used to combine two or more relations	
Assuming the files (customers.txt) 1,Ramesh,32,Ahmedabad,2000.00 2,Suresh,25,Delhi,1500.00 3,kuresh,23,Kota,2000.00 4,Kalesh,25,Mumbai,6500.00 5,Sailesh,27,Bhopal,8500.00 6,Komal,22,MP,4500.00 7,Dinesh,24,Indore,10000.00	Order.txt 102,2009-10-08 00:00:00,3,3000 100,2009-10-08 00:00:00,3,1500 101,2009-11-20 00:00:00,2,1560 103,2008-05-20 00:00:00,4,2060

grunt>customers = load '/home/cloudera/customers.txt' using PigStorage(',')as (id:int, name:chararray, age:int, address:chararray, salary:int);
grunt>orders = load 'home/cloudera/orders.txt' using PigStorage(',')as (oid:int, date:chararray, customer_id:int, amount:int);	

Self-join is used to join a table with itself as if the table were two relations.
Syntax: Relation3_name = join Relation1_name BY key, Relation2_name BY key
 	
grunt> cust_realation1 = load '/home/cloudera/customers.txt' using PigStorage(',')as (id:int, name:chararray, age:int, address:chararray, salary:int);
grunt> cust_realation2 = load '/home/cloudera/customers.txt' using PigStorage(',')as (id:int, name:chararray, age:int, address:chararray, salary:int);
grunt> customers3 = JOIN cust_relation1 BY id, cust_relation2 BY id;
 	
Inner Join	

Inner join returns rows when there is a match in both tables.

Syntax: Relation3_name = join Relation1_name BY key, Relation2_name BY key

```
grunt> cust_realation1 = load '/home/cloudera/customers.txt' using PigStorage(',')as (id:int, name:chararray, age:int, address:chararray, salary:int);

grunt> cust_realation2 = load '/home/cloudera/customers.txt' using PigStorage(',')as (id:int, name:chararray, age:int, address:chararray, salary:

grunt> customers3 = JOIN cust_relation1 BY id, cust_relation2 BY id;
```

SORT Operation

Assume the file (raw_sales.txt) with the following contents

```
CatZ,Prod22-cZ,30,60
CatA,Prod88-cA,15,50
CatY,Prod07-cY,20,40
CatB,Prod18-cB,10,50
CatX,Prod29-cZ,40,60
CatC,Prod09-cC,80,140
CatZ,Prod83-cZ,20,60
CatA,Prod17-cA,25,50
CatY,Prod98-cY,10,40
CatB,Prod99-cB,30,50
CatX,Prod19-cZ,10,60
CatC,Prod73-cC,50,140
CatZ,Prod52-cZ,10,60
CatA,Prod58-cA,15,50
CatY,Prod57-cY,10,40
CatB,Prod58-cB,10,50
CatX,Prod59-cZ,10,60
CatC,Prod59-cC,10,140
```

```
grunt> rawSales = LOAD 'raw_sales.txt' USING PigStorage(',') AS (category: chararray, product: chararray, sales: long, total_sales_category: long);
grunt> DUMP rawSales;
```

```
grpByCatTotals = GROUP rawSales BY (total_sales_category, category);
grunt> DUMP grpByCatTotals
```

```
sortGrpByCatTotals = ORDER grpByCatTotals BY group DESC;
grunt> sortGrpByCatTotals
```

```
topSalesCats = LIMIT sortGrpByCatTotals 2;
grunt> topSalesCats
```

	Perform DDL operations on Hive

DDL: Data Definition Language

1. CREATE
2. ALTER
3. DROP

CREATE TABLE

Creates a new table and specifies its characteristics.

```
hive> CREATE TABLE Employee (empid INT, empname STRING, empcity STRING);

hive> describe Employee;

hive> insert into Employee values (200,'Sreedhar','Kurnool');

hive> select * from Employee;
```

ALTER TABLE

Alter Table statement is used to alter a table in Hive.

```
hive> ALTER TABLE Employee RENAME to GPREmployee

hive> desc GPREmployee;

hive> ALTER TABLE GPREmployee ADD COLUMNS (Sal BIGINT);
```

DROP TABLE

DROP TABLE removes the table in Hive

```
hive> DROP TABLE GPREmployee;

hive> desc GPREmployee
```

08. Implementation of data management using NOSQL databases.

HBASE:

HBase is a column oriented database management system derived from Google’s NoSQL database BigTable that runs on top of HDFS.

Create table: Creates a table

```
hbase> create 'st_percentage', 'Rollno', 'Percentage'
```

Describe (or) **desc**: command returns the description of the table

```
hbase> desc 'st_percentage'
```

Insert: command used to insert the values into the table

```
hbase> Insert values into table: put 'st_percentage', '1001', 'Percentage:upto7thsem','98'
```

scan: command is used to view the data in table

```
hbase> scan 'st_percentage'
```

Alter: command used to make changes to an existing table

```
hbase> alter 'st_percentage','delete'=>'percentage'
```

disable: To delete a table, the table has to be disabled first using the disable command

```
hbase> disable ‘st_percentage’
```

enable: command used to enable the table

```
hbase> enable ‘st_percentage’
```

drop: command used to delete a table. Before dropping a table, it must be disabled.

```
hbase> drop 'st_percentage'
```

exists: command used to verify, whether the table is present in the database or not.

```
hbase> exists 'st_percentage'
```