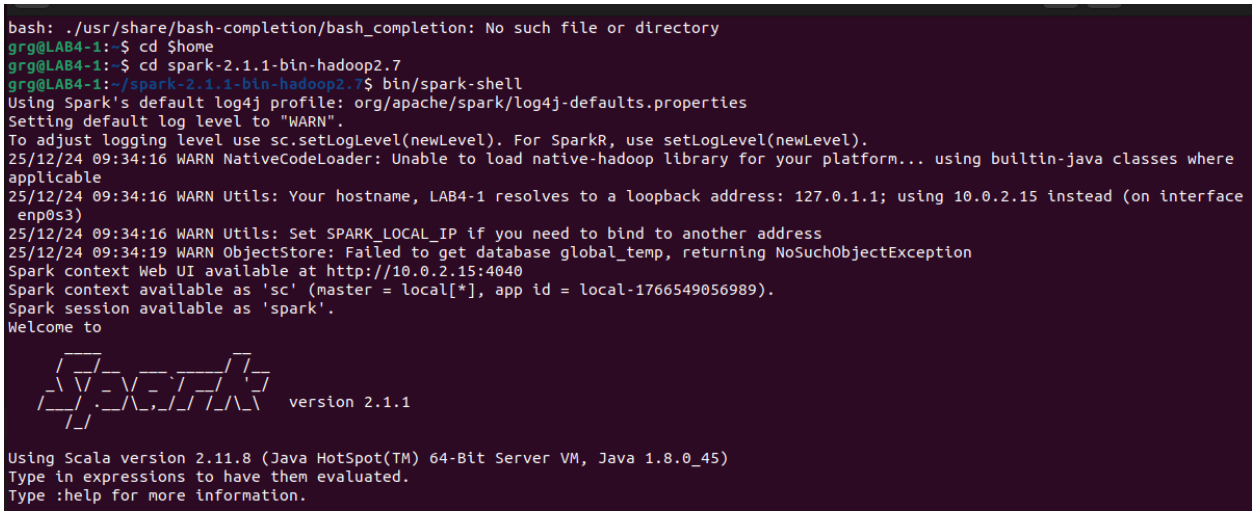
SCALA SETUP

.bashrc

export JAVA_HOME=/home/grg/jdk1.8.0_45
export PIG_HOME=/home/grg/pig-0.12.0
export PATH=$JAVA_HOME/bin:$PATH/bin:$PIG_HOME
export HADOOP_HOME=/home/grg/hadoop-2.9.1
export SPARK_HOME=//home/grg/spark-2.1.1-bin-hadoop2.7
export PATH=$JAVA_HOME/bin:$HADOOP_HOME/bin:$HIVE_HOME/bin:$PATH
export HIVE_HOME=/home/grg/hive-0.12.0
#export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PIG_HOME=/home/grg/pig-0.12.0
export PATH=$PIG_HOME/bin:$JAVA_HOME/bin:$PATH

IN TERMINAL

cd $home
cd spark-2.1.1-bin-hadoop2.7
bin/spark-shell



10) SCALA CLASS EMPLOYEE

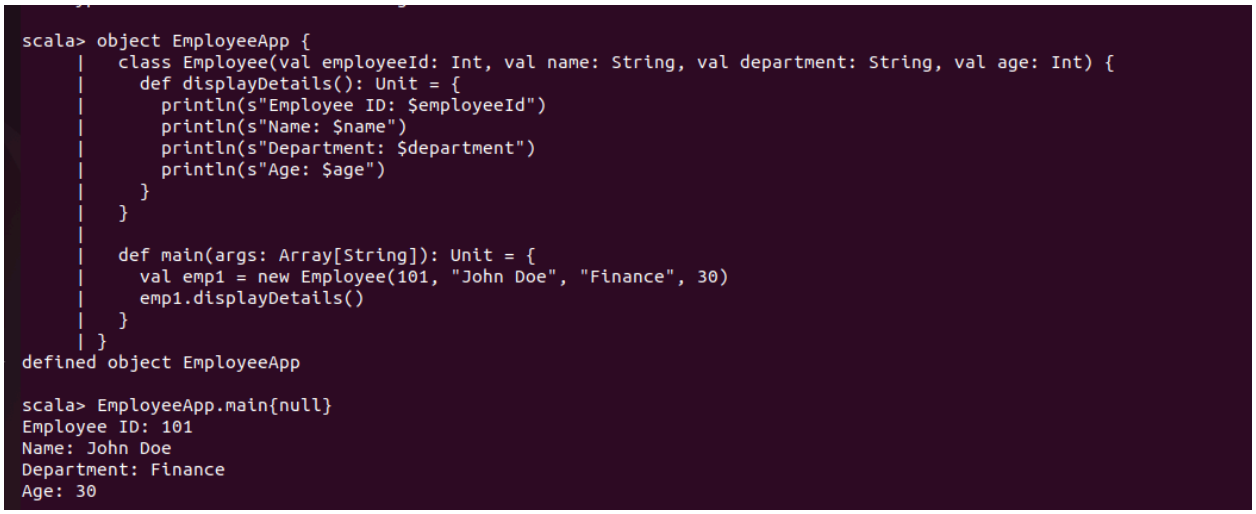
```
object EmployeeApp {
  class Employee(val employeeId: Int, val name: String, val department: String, val age: Int) {
    def displayDetails(): Unit = {
      println(s"Employee ID: $employeeId")
```

```
    println(s"Name: $name")
    println(s"Department: $department")
    println(s"Age: $age")
  }
 }

 def main(args: Array[String]): Unit = {
  val emp1 = new Employee(101, "John Doe", "Finance", 30)
  emp1.displayDetails()
 }
}
```

To view the Output:

EmployeeApp.main{null}

OUTPUT:

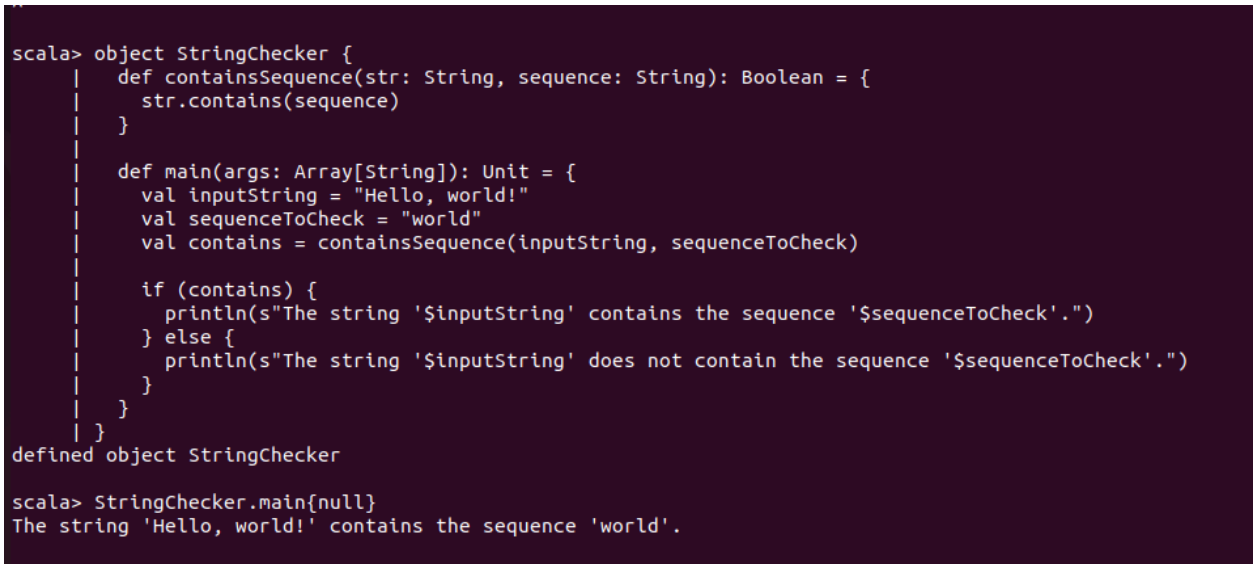


11) SCALA STRING CHECKER

```
object StringChecker {
 def containsSequence(str: String, sequence: String): Boolean = {
  str.contains(sequence)
 }

 def main(args: Array[String]): Unit = {
  val inputString = "Hello, world!"
```

```
    val sequenceToCheck = "world"
    val contains = containsSequence(inputString, sequenceToCheck)

    if (contains) {
      println(s"The string '$inputString' contains the sequence '$sequenceToCheck'.")
    } else {
      println(s"The string '$inputString' does not contain the sequence '$sequenceToCheck'.")
    }
  }
}
```

OUTPUT:



12) SCALA MIN AND MAX VALUES

```
object MapMinMax {
  def main(args: Array[String]): Unit = {
    val numbersMap = Map("one" -> 1, "two" -> 2, "three" -> 3, "four" -> 4, "five" -> 5)
    val minValue = numbersMap.minBy(_._2)  // Get the pair with the minimum value
    val maxValue = numbersMap.maxBy(_._2)  // Get the pair with the maximum value
    println(s"Minimum Value in Map: ${minValue._2}")
    println(s"Maximum Value in Map: ${maxValue._2}")
  }
}
```

To View the Output:

MapMinMax.main{null}

OUTPUT:

```
scala> object MapMinMax {
     |   def main(args: Array[String]): Unit = {
     |     val numbersMap = Map("one" -> 1, "two" -> 2, "three" -> 3, "four" -> 4, "five" -> 5)
     |     val minValue = numbersMap.minBy(_._2)  // Get the pair with the minimum value
     |     val maxValue = numbersMap.maxBy(_._2)  // Get the pair with the maximum value
     |     println(s"Minimum Value in Map: ${minValue._2}")
     |     println(s"Maximum Value in Map: ${maxValue._2}")
     |   }
     | }
defined object MapMinMax

scala> MapMinMax.main{null}
Minimum Value in Map: 1
Maximum Value in Map: 5
```

13) SCALA DATA FRAME AND PERFORM GROUP

It can need **olympics.csv** file

val df2 = spark.read.option("inferSchema", "true").option("delimiter", ",").option("header", "true").csv("/home/grg/olymbics.csv")
df2.filter($"Medal" === "Silver" || $"Medal" === "Gold").groupBy($"City").count().show()
df2.groupBy($"City", $"Medal", $"Year").count().show()
df2.groupBy($"City", $"Medal", $"Sex").count().show()
df2.filter($"Sex" === "F").groupBy($"City", $"Medal").count().show()

OUTPUT:

```
scala> df2.filter($"Medal" === "Silver" || $"Medal" === "Gold").groupBy($"City").count().show()
+------------+-----+
|        City|count|
+------------+-----+
|     Beijing| 1338|
|    Grenoble|  136|
|   Stockholm|  664|
|Squaw Valley|   93|
| Los Angeles| 1393|
|    Sarajevo|  148|
|        Oslo|   86|
|      Berlin|  613|
|       Sochi|  398|
|    Chamonix|   93|
|      London| 2399|
|   St. Louis|  332|
|      Sydney| 1323|
|       Tokyo|  677|
|    Montreal|  866|
| Mexico City|  692|
|       Paris|  981|
| Albertville|  211|
|        Roma|  596|
| Lillehammer|  218|
+------------+-----+
only showing top 20 rows
```

```
scala> df2.groupBy($"City", $"Medal", $"Year").count().show()
+-----------+--------------------+-----------+-----+
|       City|               Medal|       Year|count|
+-----------+--------------------+-----------+-----+
|     Summer|Rowing Men's Sing...|1952 Summer|    1|
|     Summer|Canoeing Men's Ka...|1964 Summer|    1|
|     Summer|Wrestling Men's L...|1960 Summer|    1|
|     Summer|Athletics Men's 1...|2012 Summer|    1|
|     Summer|Swimming Women's ...|1936 Summer|    1|
|     Summer|Athletics Men's 4...|1968 Summer|    1|
|     Winter|Speed Skating Wom...|1976 Winter|    1|
|     Summer|Athletics Men's 4...|1988 Summer|    1|
|   Grenoble|              Silver|       1968|   70|
|     Summer|Football Men's Fo...|1912 Summer|    1|
|     Summer|Athletics Men's D...|1968 Summer|    1|
|     Winter|Alpine Skiing Wom...|1952 Winter|    1|
|     Summer|Swimming Men's 20...|1996 Summer|    1|
|     Summer|Athletics Women's...|1992 Summer|    1|
|     Summer|Equestrianism Men...|1952 Summer|    2|
|Albertville|              Bronze|       1992|  106|
|     Summer|Equestrianism Men...|1912 Summer|    1|
|     Summer|Cycling Men's Tea...|1956 Summer|    1|
|     Summer|Athletics Men's D...|1960 Summer|    1|
|     Summer|Athletics Men's 1...|1948 Summer|    1|
+-----------+--------------------+-----------+-----+
only showing top 20 rows
```

```
scala> df2.groupBy($"City", $"Medal", $"Sex").count().show()
+------------------+------------------+------------+-----+
|              City|             Medal|         Sex|count|
+------------------+------------------+------------+-----+
|           Beijing|            Silver|           M|  360|
|          Montreal|            Bronze|           M|  317|
|              Oslo|            Silver|           F|    6|
|            Winter|Luge Mixed (Men)'...|        Jr."|    3|
|            Summer|Handball Men's Ha...|        Jr."|    3|
|              1900|              Golf|      Wright-|    1|
|            Summer|Boxing Men's Ligh...|        Jr."|    1|
|         Vancouver|                NA|           F| 1618|
|       Mexico City|            Bronze|           M|  285|
|            Summer|Equestrianism Mix...|        III"|    1|
|            Summer|Swimming Women's ...|     -Greer)"|    1|
|    Rio de Janeiro|            Silver|           M|  335|
|            Moskva|                NA|           F| 1322|
|Garmisch-Partenki...|            Silver|           F|    3|
|            Summer|Gymnastics Women'...| -Zuschneid)"|    1|
|       Los Angeles|                NA|           M| 8047|
|         Stockholm|              Gold|           M|  328|
|         Melbourne|                NA|           M| 3228|
|            Berlin|            Bronze|           M|  263|
|            Summer|Rowing Men's Doub...|        Jr."|    5|
+------------------+------------------+------------+-----+
only showing top 20 rows
```

```
scala> df2.filter($"Sex" === "F").groupBy($"City", $"Medal").count().show()
+------------------+------+-----+
|              City| Medal|count|
+------------------+------+-----+
|      Sankt Moritz|    NA|  143|
|       Mexico City|  Gold|   75|
|      Sankt Moritz|  Gold|    7|
|       Lillehammer|Bronze|   35|
|         St. Louis|  Gold|    6|
|Garmisch-Partenki...|Silver|    3|
|            London|Bronze|  354|
|         Amsterdam|    NA|  305|
|         Stockholm|Bronze|   12|
|         Antwerpen|Bronze|   14|
|           Sapporo|Bronze|   14|
|       Albertville|  Gold|   33|
|  Cortina d'Ampezzo|Silver|    9|
|             Sochi|    NA| 1758|
|             Paris|Silver|   20|
|         Stockholm|  Gold|   10|
|           Beijing|Bronze|  319|
|          Helsinki|Bronze|   46|
|             Seoul|Bronze|  191|
|            Moskva|    NA| 1322|
+------------------+------+-----+
only showing top 20 rows
```

16) SCALA SPARK RDD

It can need **word.txt** file
No means
Create new file in text editor

hi hi hi
hello hello hello
welcome

```
val a = sc.textFile("/home/grg/word.txt")
val splitdata = a.flatMap(line => line.split(" "))
val napdata = splitdata.map(word => (word, 1))
```

val reducedata = napdata.reduceByKey(_ + _)
reducedata.collect()

```
scala> val a = sc.textFile("/home/grg/word.txt")
a: org.apache.spark.rdd.RDD[String] = /home/grg/word.txt MapPartitionsRDD[36] at textFile at <console>:24

scala> val splitdata = a.flatMap(line => line.split(" "))
splitdata: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[37] at flatMap at <console>:26

scala> val napdata = splitdata.map(word => (word, 1))
napdata: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[38] at map at <console>:28

scala> val reducedata = napdata.reduceByKey(_ + _)
reducedata: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[39] at reduceByKey at <console>:30

scala> reducedata.collect()
res22: Array[(String, Int)] = Array((hello,3), (welcome,1), (hi,3))
```

or

val a = sc.textFile("/home/grg/word.txt")
val splitdata = a.flatMap(line => line.split(" "))
val napdata = splitdata.map(word => (word, 1))
val reducedata = napdata.reduceByKey(_ + _)
reducedata.collect().foreach(println)

```
scala> val a = sc.textFile("/home/grg/word.txt")
a: org.apache.spark.rdd.RDD[String] = /home/grg/word.txt MapPartitionsRDD[41] at textFile at <console>:24

scala> val splitdata = a.flatMap(line => line.split(" "))
splitdata: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[42] at flatMap at <console>:26

scala> val napdata = splitdata.map(word => (word, 1))
napdata: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[43] at map at <console>:28

scala> val reducedata = napdata.reduceByKey(_ + _)
reducedata: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[44] at reduceByKey at <console>:30

scala> reducedata.collect().foreach(println)  // Print the results to the console
(hello,3)
(welcome,1)
(hi,3)
```

20) SCALA STUDENT GRADE ANALYSIS

```
object StudentGradeAnalysis {
  case class Student(name: String, grade: Int)
  val students = List(
    Student("Alice", 85),
    Student("Bob", 72),
    Student("Charlie", 90),
    Student("David", 68),
    Student("Eva", 95),
    Student("Frank", 77),
    Student("Grace", 83)
  )
  def calculateAverageGrade(students: List[Student]): Double = {
```

```
  val totalGrades = students.map(_.grade).sum
  totalGrades.toDouble / students.size
 }
 def calculateHighestGrade(students: List[Student]): Student = {
  students.maxBy(_.grade)
 }
 def calculateLowestGrade(students: List[Student]): Student = {
  students.minBy(_.grade)
 }
 def generateGradeReport(students: List[Student]): Unit = {
  val average = calculateAverageGrade(students)
  val highest = calculateHighestGrade(students)
  val lowest = calculateLowestGrade(students)
  println(s"Total Students: ${students.size}")
  println(s"Average Grade: %.2f".format(average))
  println(s"Highest Grade: ${highest.name} with ${highest.grade}")
  println(s"Lowest Grade: ${lowest.name} with ${lowest.grade}")
  println("\nGrade Distribution:")
  students.sortBy(_.grade).foreach(student => {
   println(s"${student.name}: ${student.grade}")
  })
 }
 def main(args: Array[String]): Unit = {
  println("Student Grade Analysis\n")
  generateGradeReport(students)
 }
}
```

To view Output

StudentGradeAnalysis.main(null)

OUTPUT:

```
scala> object StudentGradeAnalysis {
     |   case class Student(name: String, grade: Int)
     |   val students = List(
     |     Student("Alice", 85),
     |     Student("Bob", 72),
     |     Student("Charlie", 90),
     |     Student("David", 68),
     |     Student("Eva", 95),
     |     Student("Frank", 77),
     |     Student("Grace", 83)
     |   )
     |   def calculateAverageGrade(students: List[Student]): Double = {
     |     val totalGrades = students.map(_.grade).sum
     |     totalGrades.toDouble / students.size
     |   }
     |   def calculateHighestGrade(students: List[Student]): Student = {
     |     students.maxBy(_.grade)
     |   }
     |   def calculateLowestGrade(students: List[Student]): Student = {
     |     students.minBy(_.grade)
     |   }
     |   def generateGradeReport(students: List[Student]): Unit = {
     |     val average = calculateAverageGrade(students)
     |     val highest = calculateHighestGrade(students)
     |     val lowest = calculateLowestGrade(students)
     |     println(s"Total Students: ${students.size}")
     |     println(s"Average Grade: %.2f".format(average))
     |     println(s"Highest Grade: ${highest.name} with ${highest.grade}")
     |     println(s"Lowest Grade: ${lowest.name} with ${lowest.grade}")
     |     println("\nGrade Distribution:")
     |     students.sortBy(_.grade).foreach(student => {
     |       println(s"${student.name}: ${student.grade}")
     |     })
     |   }
     | def main(args: Array[String]): Unit = {
     |     println("Student Grade Analysis\n")
     |     generateGradeReport(students)
     |   }
     | }
defined object StudentGradeAnalysis
```

```
scala> StudentGradeAnalysis.main(null)
Student Grade Analysis

Total Students: 7
Average Grade: 81.43
Highest Grade: Eva with 95
Lowest Grade: David with 68

Grade Distribution:
David: 68
Bob: 72
Frank: 77
Grace: 83
Alice: 85
Charlie: 90
Eva: 95
```

HIVE SETUP

.Bashrc

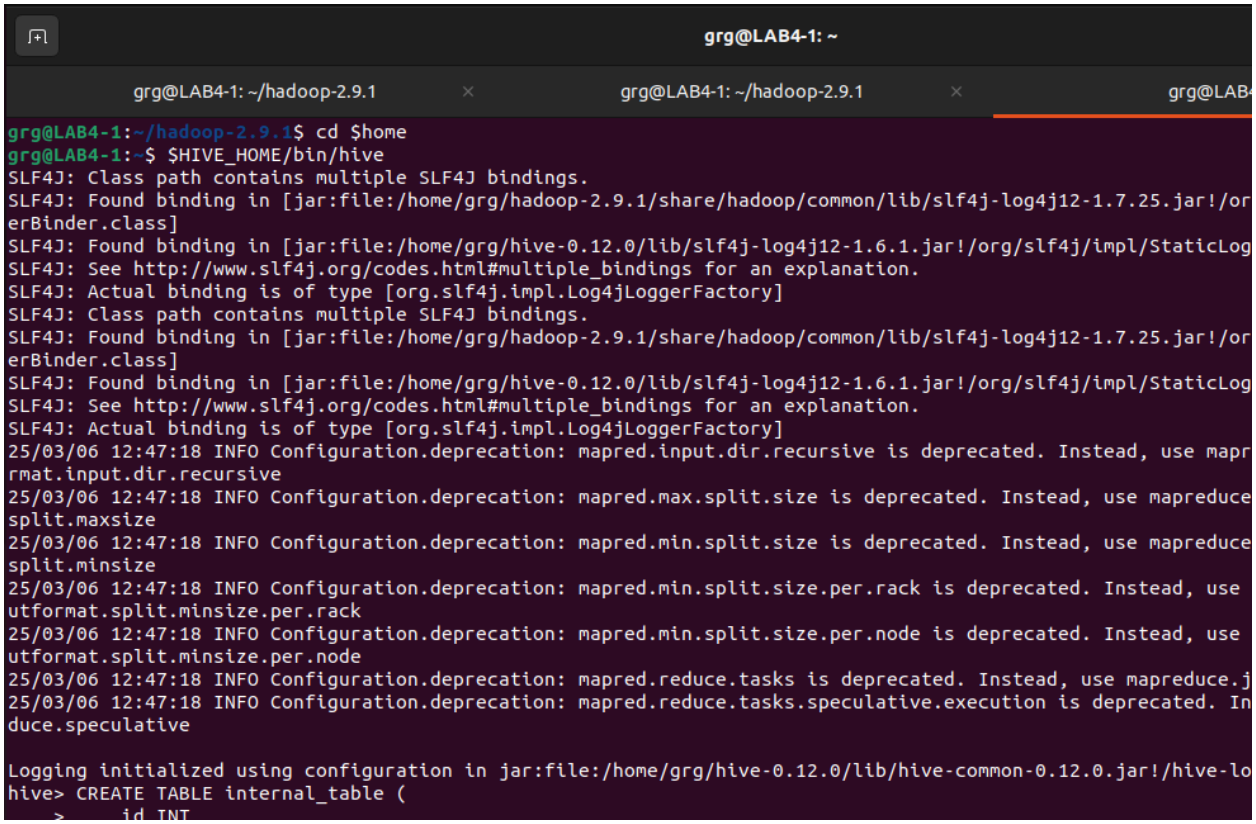
export JAVA_HOME=/home/grg/jdk1.8.0_45
export HADOOP_HOME=/home/grg/hadoop-2.9.0
export SPARK_HOME=/home/grg/spark-2.1.1-bin-hadoop2.7

export PATH=$JAVA_HOME/bin:$HADOOP_HOME/bin:$HIVE_HOME/bin:$PATH
export JAVA_HOME=/usr/lip/jpm/java-8-openjdk_amd64
export PIG_HOME=/home/grg/pig-0.12.0
export HADOOP_HOME=/home/grg/hadoop-2.9.1
export HIVE_HOME=/home/grg/hive-0.12.0
export PATH=$HIVE_HOME/bin:$PATH

```
94
95 # Add an "alert" alias for long running commands.  Use like so:
96 #   sleep 10; alert
97 alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo error)" "$(history|tail -n1|sed -e '\''s/
   ^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'')"'
98
99 # Alias definitions.
100 # You may want to put all your additions into a separate file like
101 # ~/.bash_aliases, instead of adding them here directly.
102 # See /usr/share/doc/bash-doc/examples in the bash-doc package.
103
104 if [ -f ~/.bash_aliases ]; then
105     . ~/.bash_aliases
106 fi
107
108 # enable programmable completion features (you don't need to enable
109 # this, if it's already enabled in /etc/bash.bashrc and /etc/profile
110 # sources /etc/bash.bashrc).
111 if ! shopt -oq posix; then
112     if [ -f /usr/share/bash-completion/bash_completion ]; then
113         . /usr/share/bash-completion/bash_completion
114     elif [ -f /etc/bash_completion ]; then
115         . /etc/bash_completion
116     fi
117 fi
118 export JAVA_HOME=/home/grg/jdk1.8.0_45
119 export HADOOP_HOME=/home/grg/hadoop-2.9.0
120 export SPARK_HOME=//home/grg/spark-2.1.1-bin-hadoop2.7
121 export PATH=$JAVA_HOME/bin:$HADOOP_HOME/bin:$HIVE_HOME/bin:$PATH
122 export JAVA_HOME=/usr/lip/jpm/java-8-openjdk_amd64
123 export PIG_HOME=/home/grg/pig-0.12.0
124 export HADOOP_HOME=/home/grg/hadoop-2.9.1
125 export HIVE_HOME=/home/grg/hive-0.12.0
126 export PATH=$HIVE_HOME/bin:$PATH
```

TERMINAL

cd $home
$HIVE_HOME/bin/hive



6) HIVE INTERNAL AND EXRERNAL TABLE

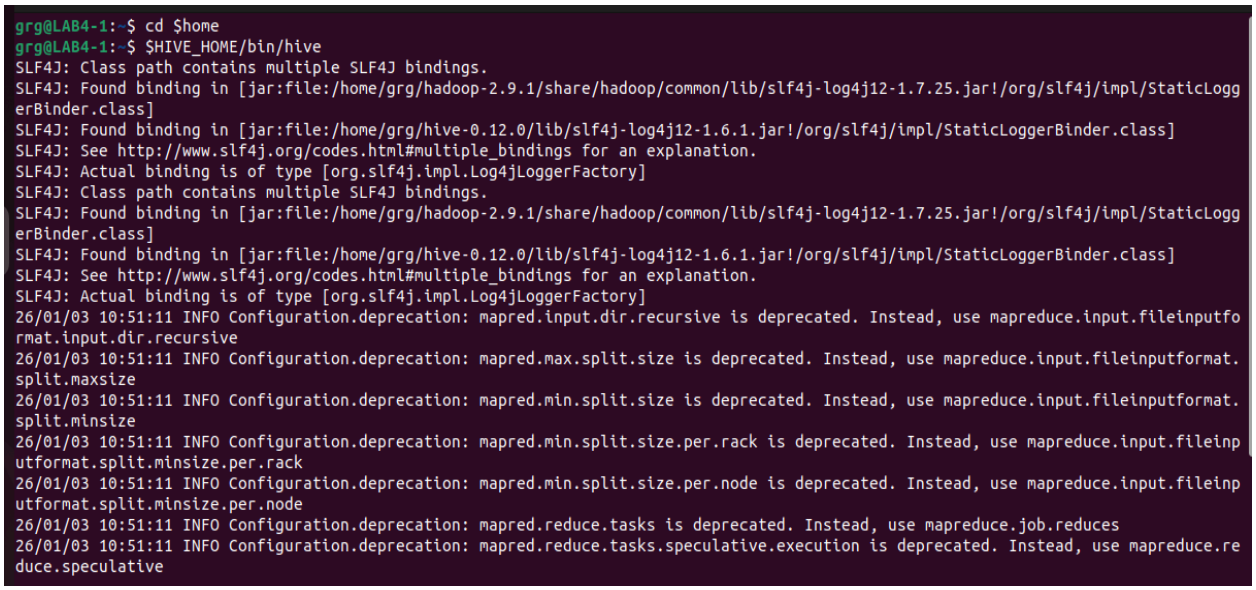
```
CREATE TABLE internal_table (
    id INT,
    name STRING,
    age INT
)
STORED AS TEXTFILE;

CREATE EXTERNAL TABLE external_table (
    id INT,
    name STRING,
    age INT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LOCATION '/home/grg/external_data/';


DESCRIBE internal_table;
```

DESCRIBE external_table;

ALTER TABLE internal_table CHANGE COLUMN age age BIGINT;
ALTER TABLE internal_table RENAME TO managed_table;
DESCRIBE managed_table;

DROP TABLE external_table;
DESCRIBE external_table;

OUTPUT:



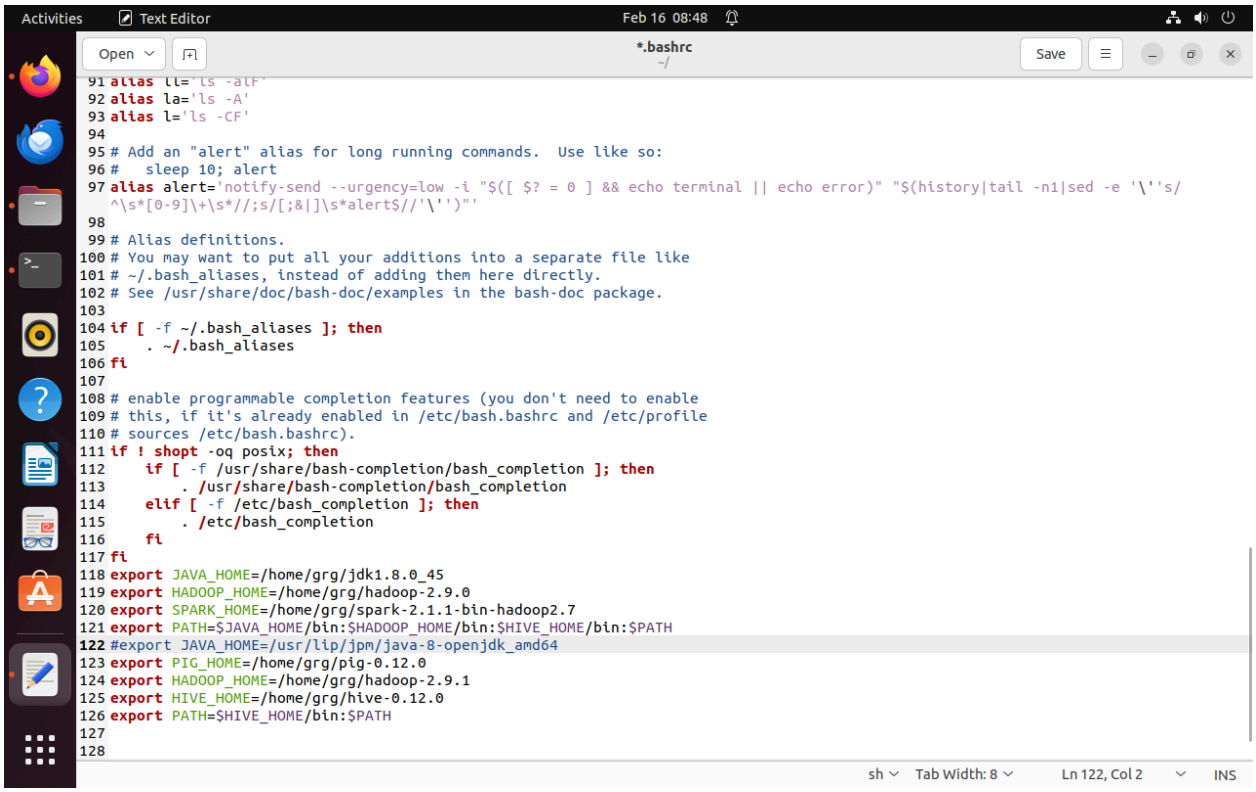
```
26/01/03 10:51:11 INFO Configuration.deprecation: mapred.reduce.tasks is deprecated. Instead, use mapreduce.job.reduces
26/01/03 10:51:11 INFO Configuration.deprecation: mapred.reduce.tasks.speculative.execution is deprecated. Instead, use mapreduce.re
duce.speculative

Logging initialized using configuration in jar:file:/home/grg/hive-0.12.0/lib/hive-common-0.12.0.jar!/hive-log4j.properties
hive> CREATE TABLE internal_table (
    >     id INT,
    >     name STRING,
    >     age INT
    > )
    > STORED AS TEXTFILE;
OK
Time taken: 3.231 seconds
hive> CREATE EXTERNAL TABLE external_table (
    >     id INT,
    >     name STRING,
    >     age INT
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LOCATION '/home/grg/external_data/';
OK
Time taken: 0.025 seconds
hive> DESCRIBE internal_table;
OK
id                      int                     None
name                    string                  None
age                     int                     None
Time taken: 0.141 seconds, Fetched: 3 row(s)
hive> DESCRIBE external_table;
OK
id                      int                     None
name                    string                  None
age                     int                     None
Time taken: 0.031 seconds, Fetched: 3 row(s)
hive> ALTER TABLE internal_table CHANGE COLUMN age age BIGINT;
OK
Time taken: 0.071 seconds
hive> ALTER TABLE internal_table RENAME TO managed_table;
```

```
Time taken: 3.231 seconds
hive> CREATE EXTERNAL TABLE external_table (
    >     id INT,
    >     name STRING,
    >     age INT
    > )
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LOCATION '/home/grg/external_data/';
OK
Time taken: 0.025 seconds
hive> DESCRIBE internal_table;
OK
id                      int                     None
name                    string                  None
age                     int                     None
Time taken: 0.141 seconds, Fetched: 3 row(s)
hive> DESCRIBE external_table;
OK
id                      int                     None
name                    string                  None
age                     int                     None
Time taken: 0.031 seconds, Fetched: 3 row(s)
hive> ALTER TABLE internal_table CHANGE COLUMN age age BIGINT;
OK
Time taken: 0.071 seconds
hive> ALTER TABLE internal_table RENAME TO managed_table;
OK
Time taken: 0.198 seconds
hive> DESCRIBE managed_table;
OK
id                      int                     None
name                    string                  None
age                     bigint                  None
Time taken: 0.051 seconds, Fetched: 3 row(s)
hive> DROP TABLE external_table;
OK
Time taken: 0.395 seconds
hive>
```

ZEPPELIN SETUP

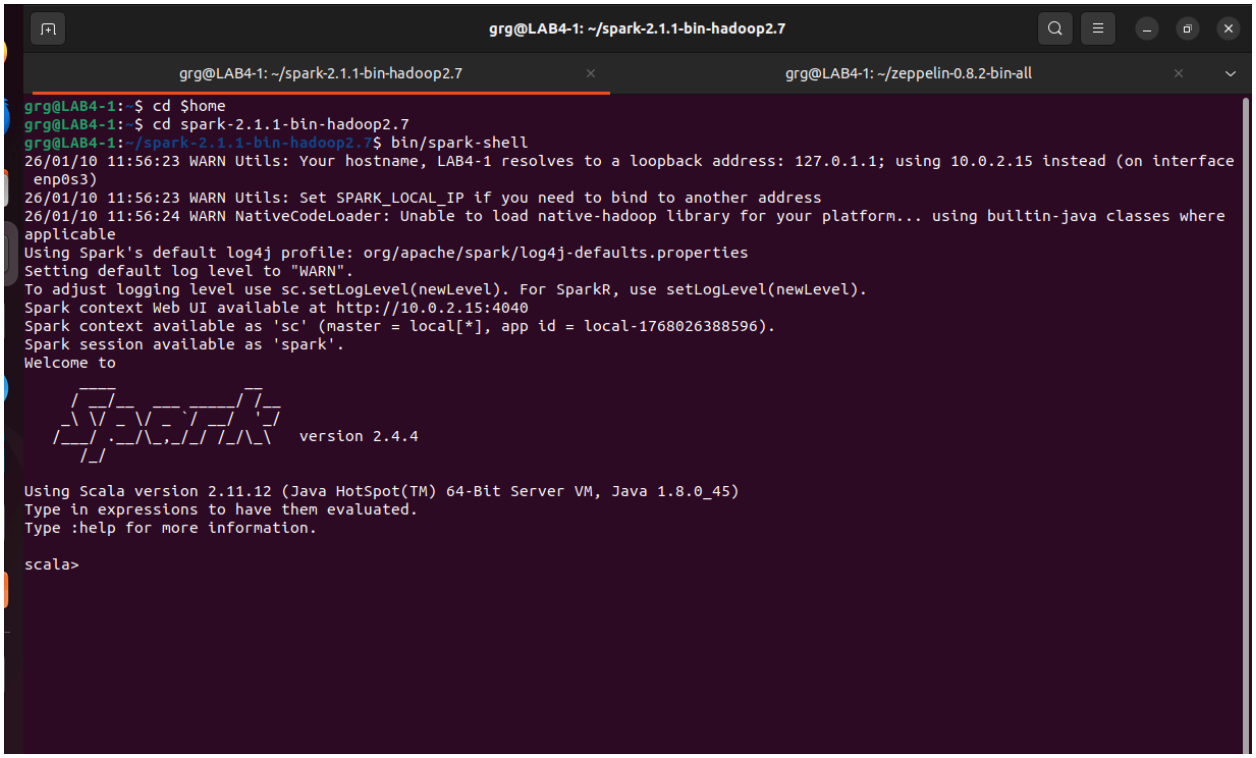
.bashrc

export JAVA_HOME=/home/grg/jdk1.8.0_45
export HADOOP_HOME=/home/grg/hadoop-2.9.0
export SPARK_HOME=/home/grg/spark-2.1.1-bin-hadoop2.7
export PATH=$JAVA_HOME/bin:$HADOOP_HOME/bin:$HIVE_HOME/bin:$PATH
#export JAVA_HOME=/usr/lip/jpm/java-8-openjdk_amd64
export PIG_HOME=/home/grg/pig-0.12.0
export HADOOP_HOME=/home/grg/hadoop-2.9.1
export HIVE_HOME=/home/grg/hive-0.12.0
export PATH=$HIVE_HOME/bin:$PATH



IN TERMINAL

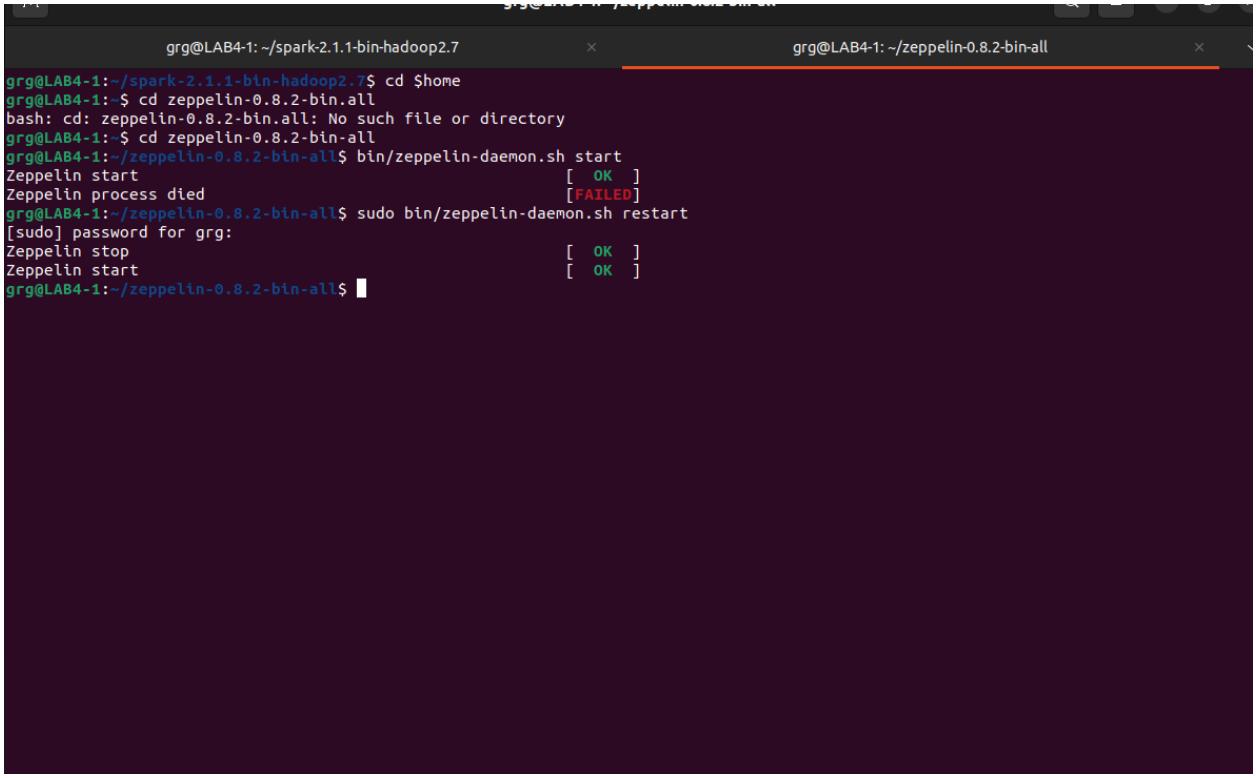
cd $home
cd spark-2.1.1-bin-hadoop2.7
bin/spark-shell



OPEN ANOTHER TERMINAL

cd $home
cd zeppelin-0.8.2-bin.all
bin/zeppelin-daemon.sh start
sudo bin/zeppelin-daemon.sh restart

Zeppelin Password : GRG@123



search firefox :http://localhost:8080/

18 ZEPPELIN CYBER CRIME DATA

create **crimedata.txt** — create new file in texteditor

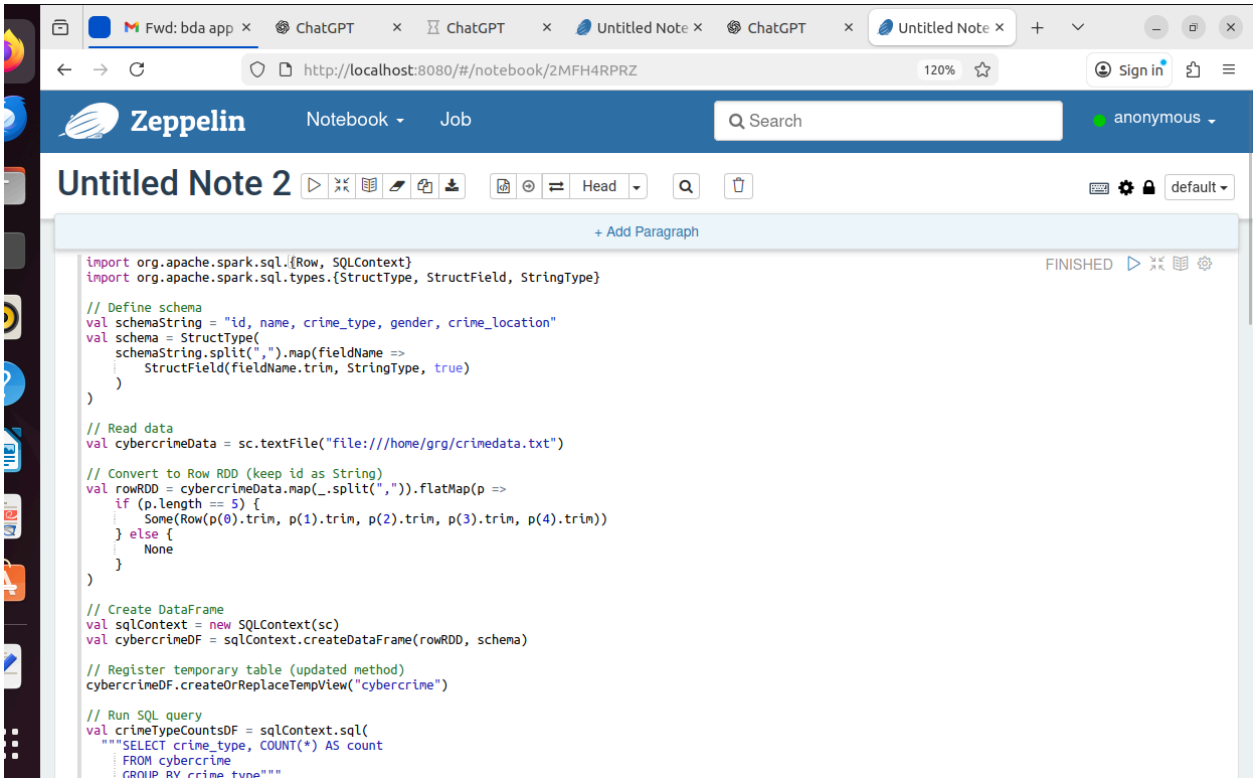
1,John,Phishing,Male,Delhi
2,Alice,Hacking,Female,Mumbai
3,Rahul,Identity Theft,Male,Bangalore
4,Sneha,Phishing,Female,Delhi
5,Amit,Online Fraud,Male,Chennai
6,Priya,Hacking,Female,Bangalore
7,Rohit,Phishing,Male,Mumbai
8,Neha,Online Fraud,Female,Delhi
9,Karan,Identity Theft,Male,Pune
10,Anjali,Hacking,Female,Chennai

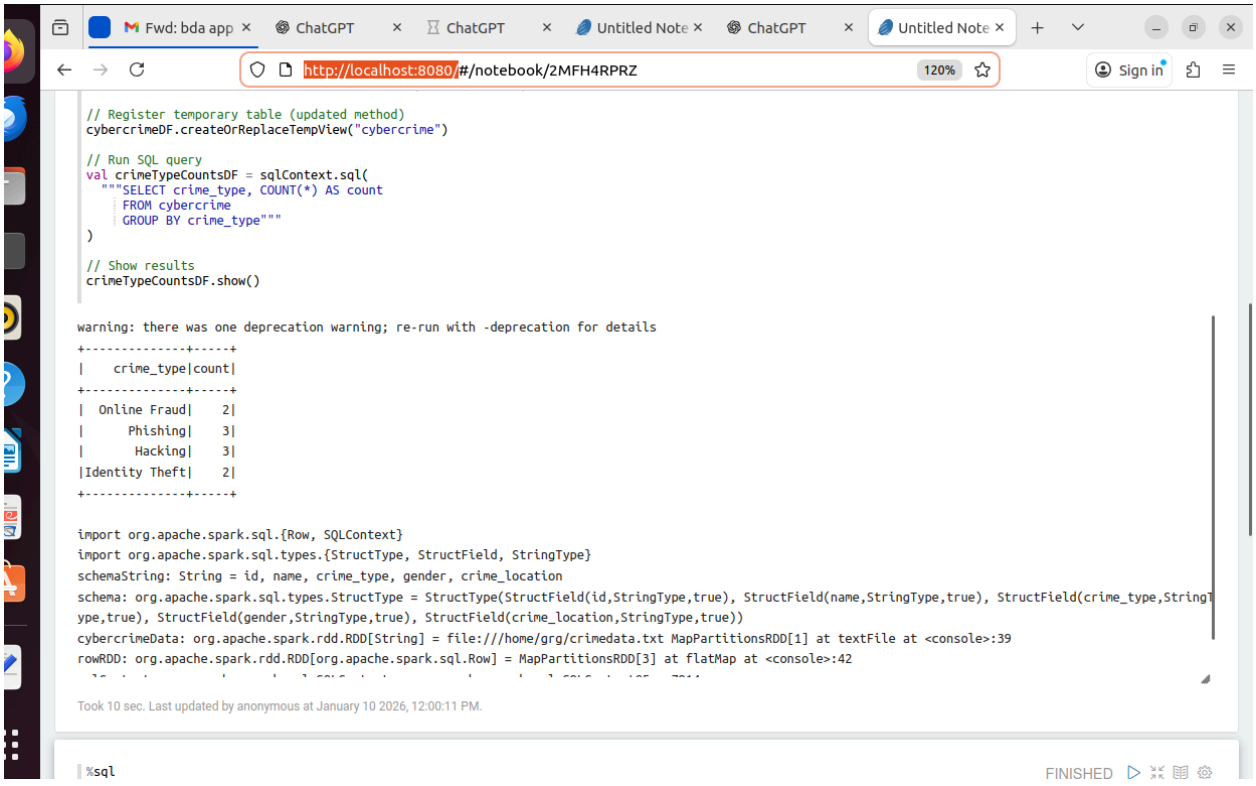
Code

```
import org.apache.spark.sql.{Row, SQLContext};
import org.apache.spark.sql.types.{StructType, StructField, StringType}
```

```
val schemaString = "id, name, crime_type, gender, crime_location"
val schema = StructType(
   schemaString.split(",").map(fieldName =>
      StructField(fieldName.trim, StringType, true)
   )
)

val cybercrimeData = sc.textFile("file:///home/grg/crimedata.txt")

val rowRDD = cybercrimeData.map(_.split(",")).flatMap(p =>
   if (p.length == 5) {
      Some(Row(p(0).trim, p(1).trim, p(2).trim, p(3).trim, p(4).trim))
   } else {
      None
   }
)

val sqlContext = new SQLContext(sc)
val cybercrimeDF = sqlContext.createDataFrame(rowRDD, schema)

cybercrimeDF.createOrReplaceTempView("cybercrime")

val crimeTypeCountsDF = sqlContext.sql(
  """SELECT crime_type, COUNT(*) AS count
     FROM cybercrime
     GROUP BY crime_type"""
)

crimeTypeCountsDF.show()
```

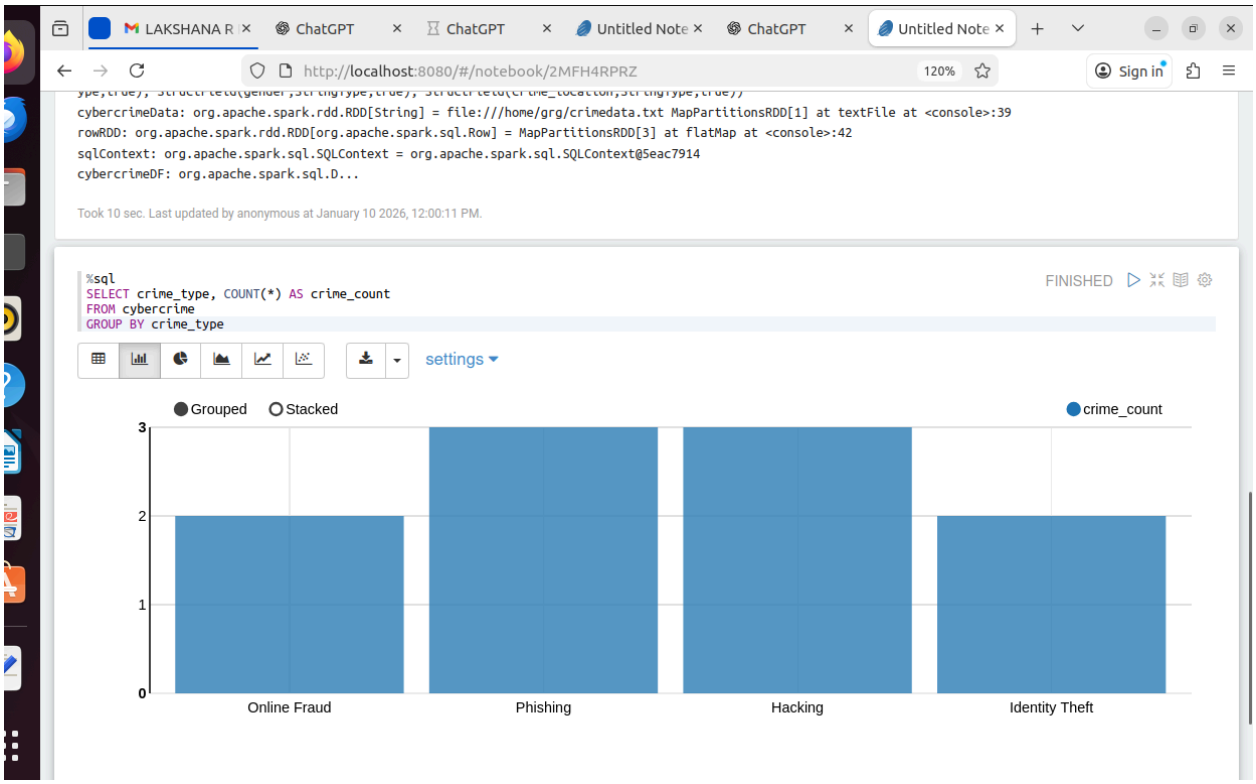
OUTPUT:


Fwd: bda app
ChatGPT
ChatGPT
Untitled Note
ChatGPT
Untitled Note
http://localhost:8080/#/notebook/2MFH4RPRZ
120%
Sign in
Zeppelin
Notebook
Job
Search
anonymous
Untitled Note 2
Head
default
+ Add Paragraph
FINISHED
import org.apache.spark.sql.{Row, SQLContext}
import org.apache.spark.sql.types.{StructType, StructField, StringType}
// Define schema
val schemaString = "id, name, crime_type, gender, crime_location"
val schema = StructType(
schemaString.split(",").map(fieldName =>
StructField(fieldName.trim, StringType, true)
)
)
// Read data
val cybercrimeData = sc.textFile("file:///home/grg/crimedata.txt")
// Convert to Row RDD (keep id as String)
val rowRDD = cybercrimeData.map(_.split(",")).flatMap(p =>
if (p.length == 5) {
Some(Row(p(0).trim, p(1).trim, p(2).trim, p(3).trim, p(4).trim))
} else {
None
}
)
// Create DataFrame
val sqlContext = new SQLContext(sc)
val cybercrimeDF = sqlContext.createDataFrame(rowRDD, schema)
// Register temporary table (updated method)
cybercrimeDF.createOrReplaceTempView("cybercrime")
// Run SQL query
val crimeTypeCountsDF = sqlContext.sql(
"""SELECT crime_type, COUNT(*) AS count
FROM cybercrime
GROUP BY crime_type"""


Fwd: bda app
ChatGPT
ChatGPT
Untitled Note
ChatGPT
Untitled Note
http://localhost:8080/#/notebook/2MFH4RPRZ
120%
Sign in
// Register temporary table (updated method)
cybercrimeDF.createOrReplaceTempView("cybercrime")
// Run SQL query
val crimeTypeCountsDF = sqlContext.sql(
"""SELECT crime_type, COUNT(*) AS count
FROM cybercrime
GROUP BY crime_type"""
)
// Show results
crimeTypeCountsDF.show()
warning: there was one deprecation warning; re-run with -deprecation for details
+--------------+-----+
| crime_type|count|
+--------------+-----+
Online Fraud	2
Phishing	3
Hacking	3
Identity Theft	2
+--------------+-----+
import org.apache.spark.sql.{Row, SQLContext}
import org.apache.spark.sql.types.{StructType, StructField, StringType}
schemaString: String = id, name, crime_type, gender, crime_location
schema: org.apache.spark.sql.types.StructType = StructType(StructField(id,StringType,true), StructField(name,StringType,true), StructField(crime_type,StringT
ype,true), StructField(gender,StringType,true), StructField(crime_location,StringType,true))
cybercrimeData: org.apache.spark.rdd.RDD[String] = file:///home/grg/crimedata.txt MapPartitionsRDD[1] at textFile at <console>:39
rowRDD: org.apache.spark.rdd.RDD[org.apache.spark.sql.Row] = MapPartitionsRDD[3] at flatMap at <console>:42
Took 10 sec. Last updated by anonymous at January 10 2026, 12:00:11 PM.
%sql
FINISHED

To view the Output:

%sql
SELECT crime_type, COUNT(*) AS crime_count
FROM cybercrime
GROUP BY crime_type



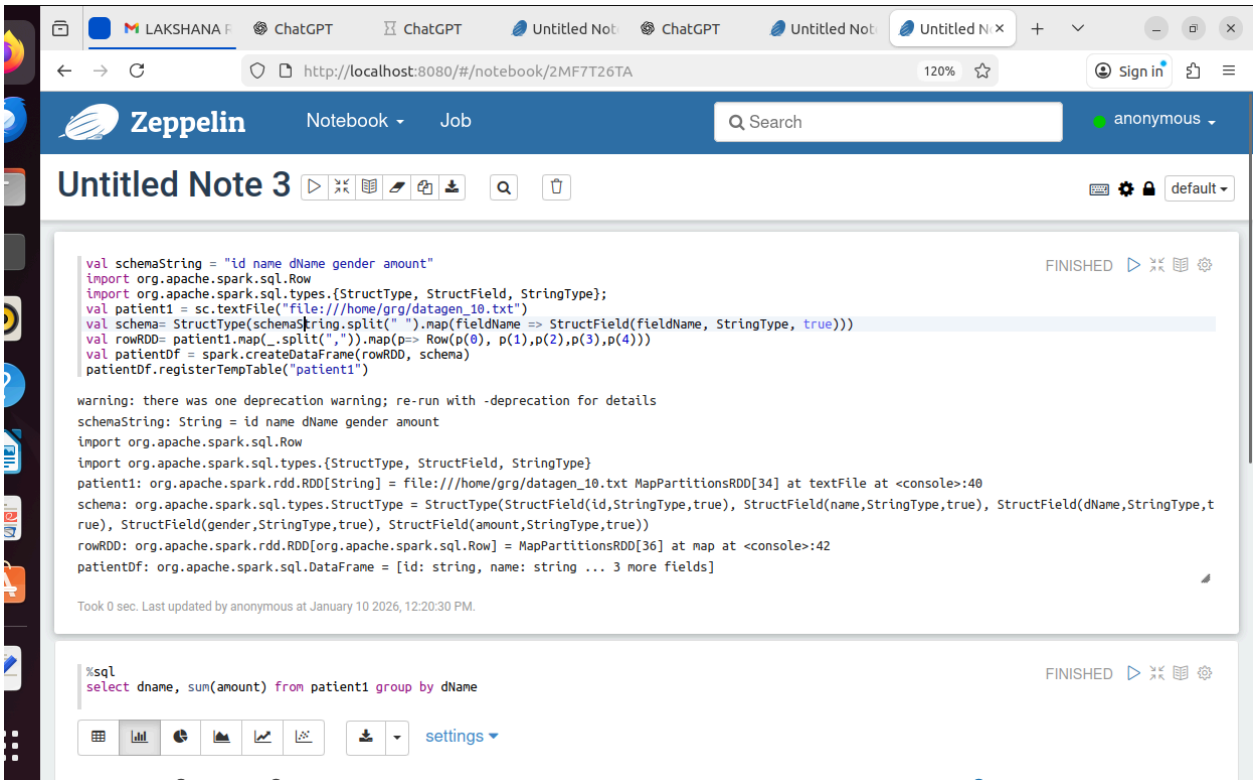
19–zeppelin

It can need **datagen_10.txt**
No means
Create text editor
1,Brandon Buckner,avil,female,525
2,Veda Hopkins,avil,male,633
3,Zia Underwood,paracetamol,male,980
4,Austin Mayer,paracetamol,female,338
5,Mara Higgins,avil,female,153
6,Sybill Crosby,avil,male,193
7,Tyler Rosales,paracetamol,male,778
8,Ivan Hale,avil,female,454
9,Alika Gilmore,paracetamol,female,833

10,Len Burgess,metacin,male,325

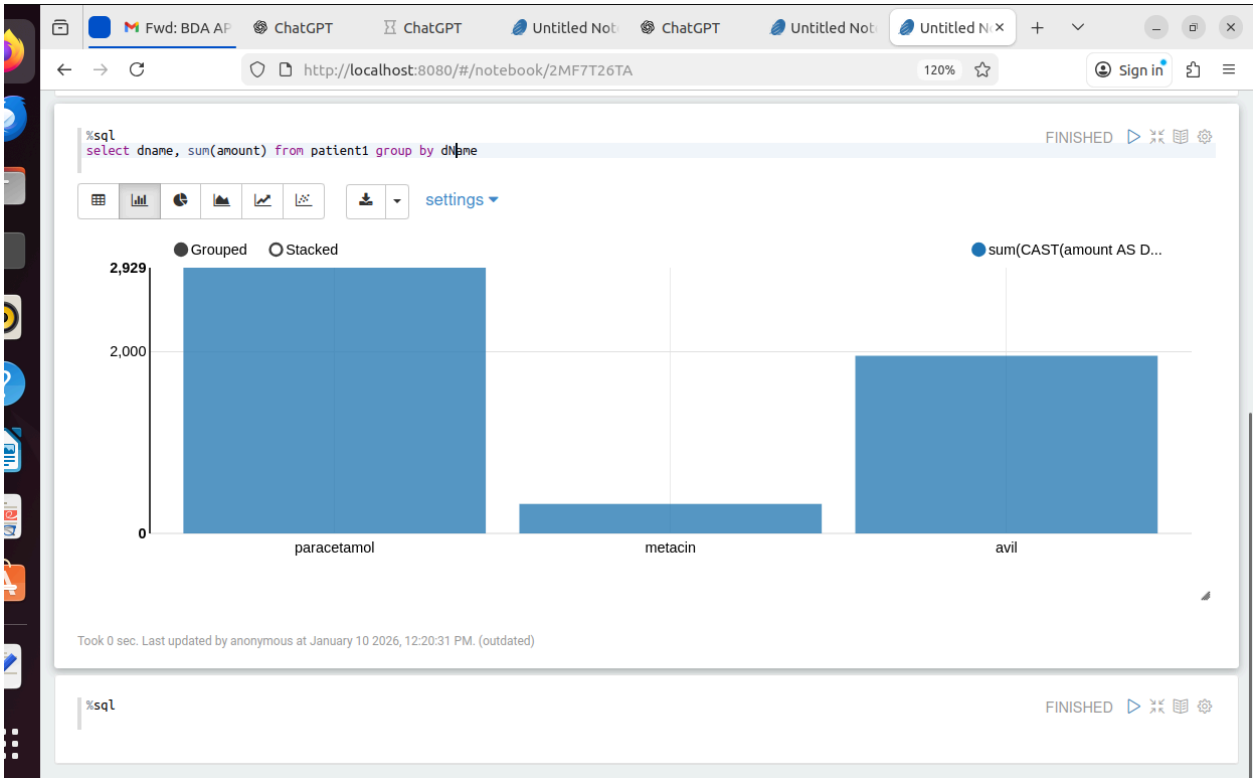
```
val schemaString = "id name dName gender amount"
import org.apache.spark.sql.Row
import org.apache.spark.sql.types.{StructType, StructField, StringType};
val patient1 = sc.textFile("file:///home/grg/datagen_10.txt")
val schema= StructType(schemaString.split(" ").map(fieldName => StructField(fieldName, StringType, true)))
val rowRDD= patient1.map(_.split(",")).map(p=> Row(p(0), p(1),p(2),p(3),p(4)))
val patientDf = spark.createDataFrame(rowRDD, schema)
patientDf.registerTempTable("patient1")
```

Output:



To view the Output

```
%sql
select dname, sum(amount) from patient1 group by dName
```

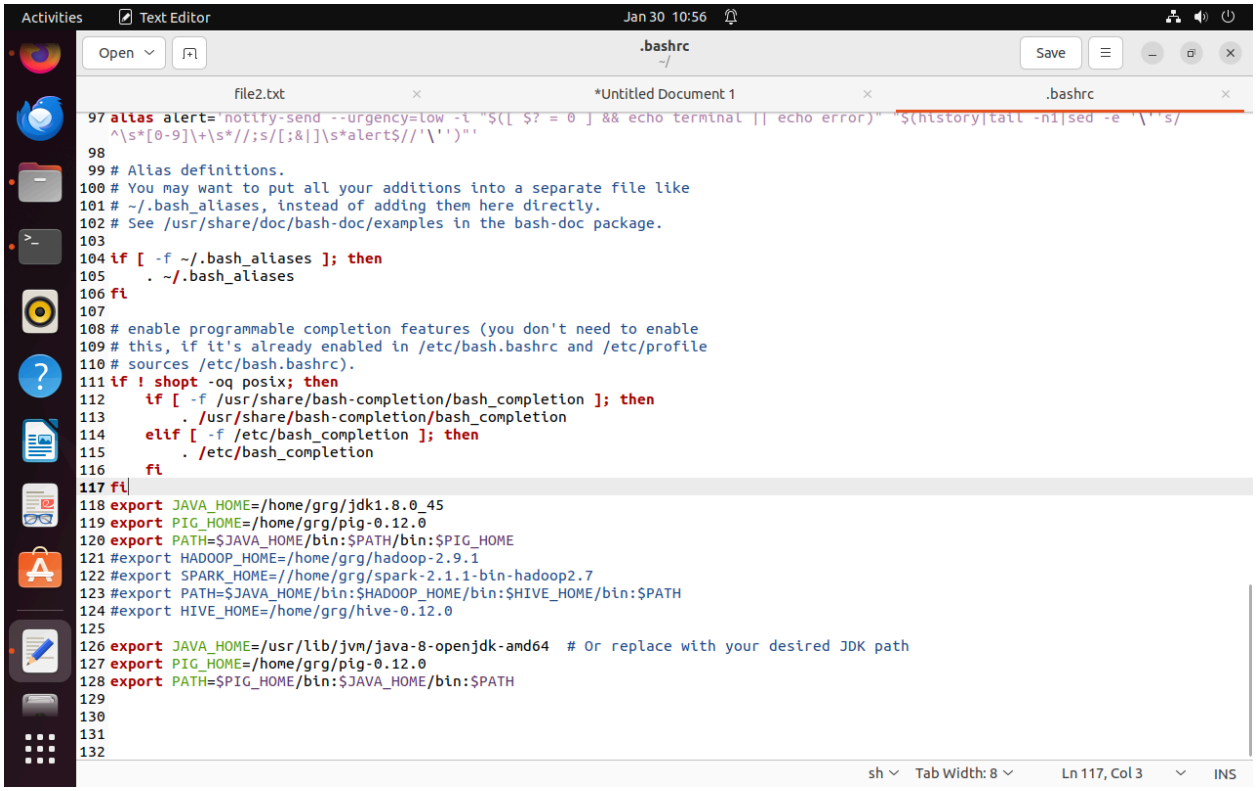

Fwd: BDA AP
ChatGPT
ChatGPT
Untitled Not
ChatGPT
Untitled Not
Untitled N
http://localhost:8080/#/notebook/2MF7T26TA
120%
Sign in
%sql
select dname, sum(amount) from patient1 group by dName
FINISHED
settings
Grouped
Stacked
sum(CAST(amount AS D...
2,929
2,000
0
paracetamol
metacin
avil
Took 0 sec. Last updated by anonymous at January 10 2026, 12:20:31 PM. (outdated)
%sql
FINISHED

PIG SETUP

.bashrc

export JAVA_HOME=/home/grg/jdk1.8.0_45
export PIG_HOME=/home/grg/pig-0.12.0
export PATH=$JAVA_HOME/bin:$PATH/bin:$PIG_HOME
#export HADOOP_HOME=/home/grg/hadoop-2.9.1
#export SPARK_HOME=//home/grg/spark-2.1.1-bin-hadoop2.7
#export PATH=$JAVA_HOME/bin:$HADOOP_HOME/bin:$HIVE_HOME/bin:$PATH
#export HIVE_HOME=/home/grg/hive-0.12.0

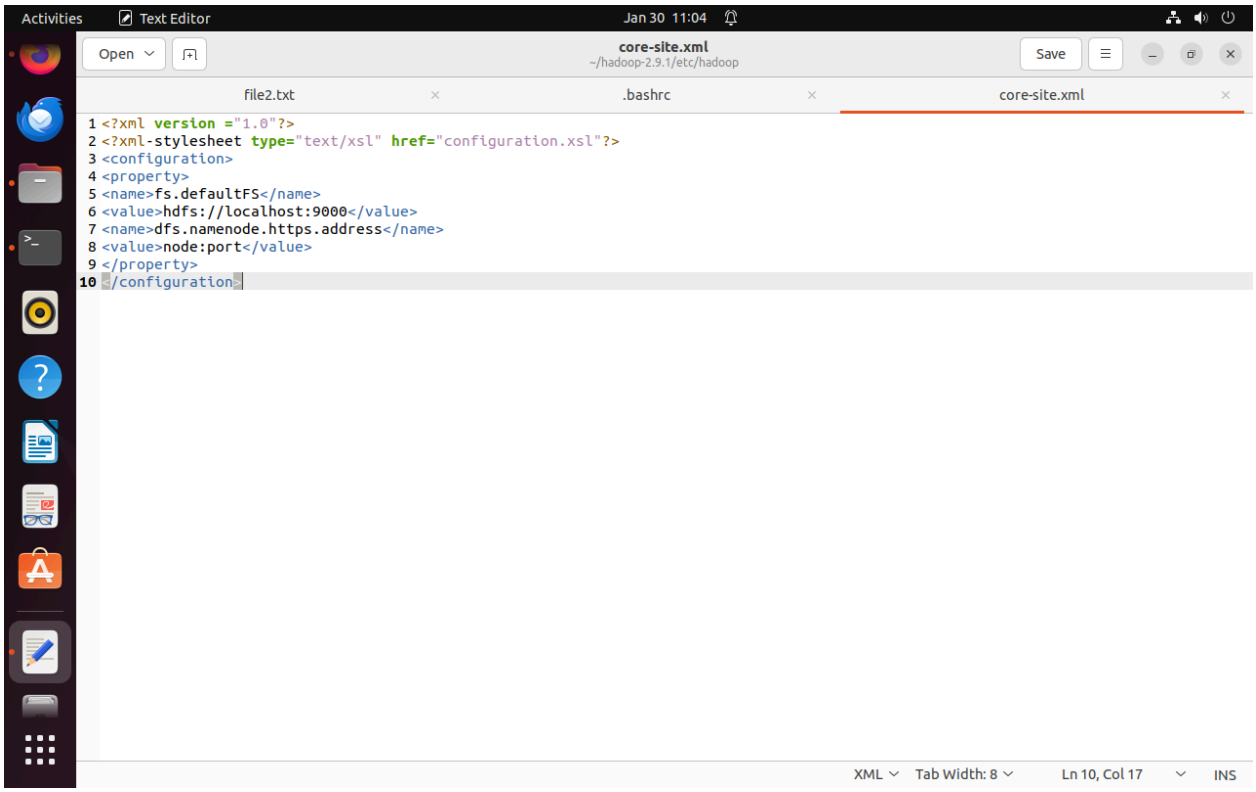
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64 # Or replace with your desired JDK path
export PIG_HOME=/home/grg/pig-0.12.0
export PATH=$PIG_HOME/bin:$JAVA_HOME/bin:$PATH



Core-site.xml

>>home/grg/hadoop-2.9.1/etc/hadoop/core.site-xml

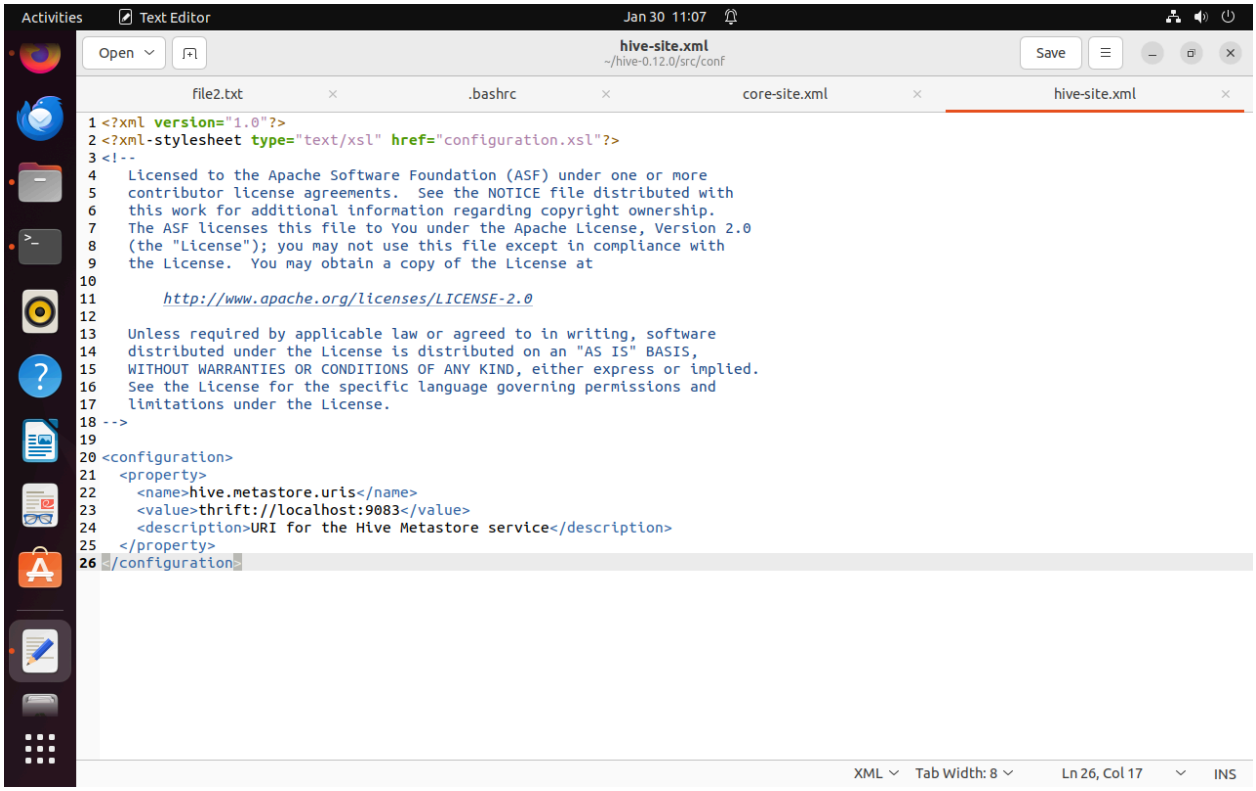
<?xml version ="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://localhost:9000</value>
<name>dfs.namenode.https.address</name>
<value>node:port</value>
</property>
</configuration>



Hive-site.xml

<configuration>
<property>
<name>hive.metastore.uris</name>

<value>thrift://localhost:9083</value>
<description>URI for the Hive Metastore service</description>
</property>
</configuration>



TERMINAL:

source .bashrc
cd pig-0.12.0
bin/pig -x local

```
grg@LAB4-1:~$ source .bashrc
grg@LAB4-1:~$ cd pig-0.12.0
grg@LAB4-1:~/pig-0.12.0$ bin/pig -x local
2026-01-30 10:44:34,870 [main] INFO  org.apache.pig.Main - Apache Pig version 0.12.0 (r1529718) compiled Oct 07 2013, 12:20:14
2026-01-30 10:44:34,871 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/grg/pig-0.12.0/pig_1769750074866.log
2026-01-30 10:44:34,896 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file /home/grg/.pigbootup not found
2026-01-30 10:44:35,059 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file syst
em at: file:///
grunt> H = LOAD'/home/user/file2.txt'using PigStorage(',')as (id: int,
>> name:chararray, gender: chararray, hobby: chararray);
```

4) pig script gender and hobbies

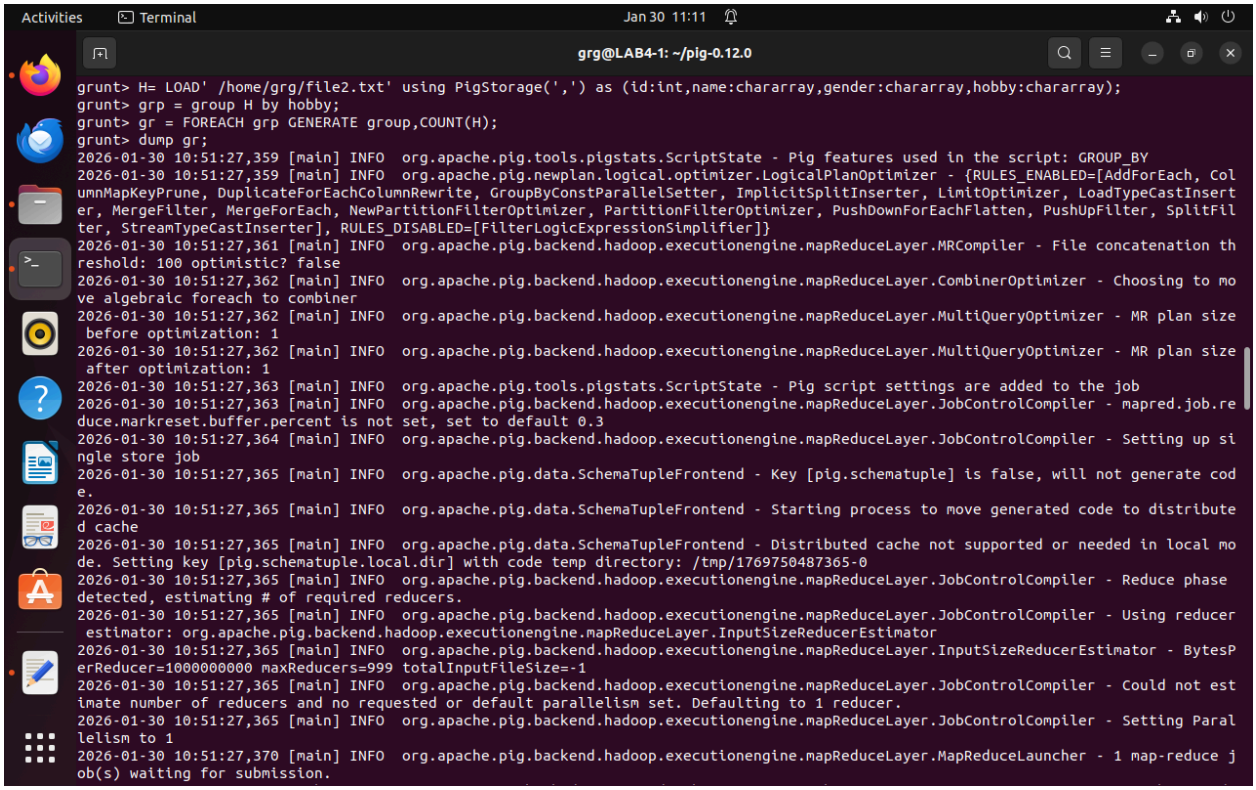
Create **file2.txt** —create new file in texteditor

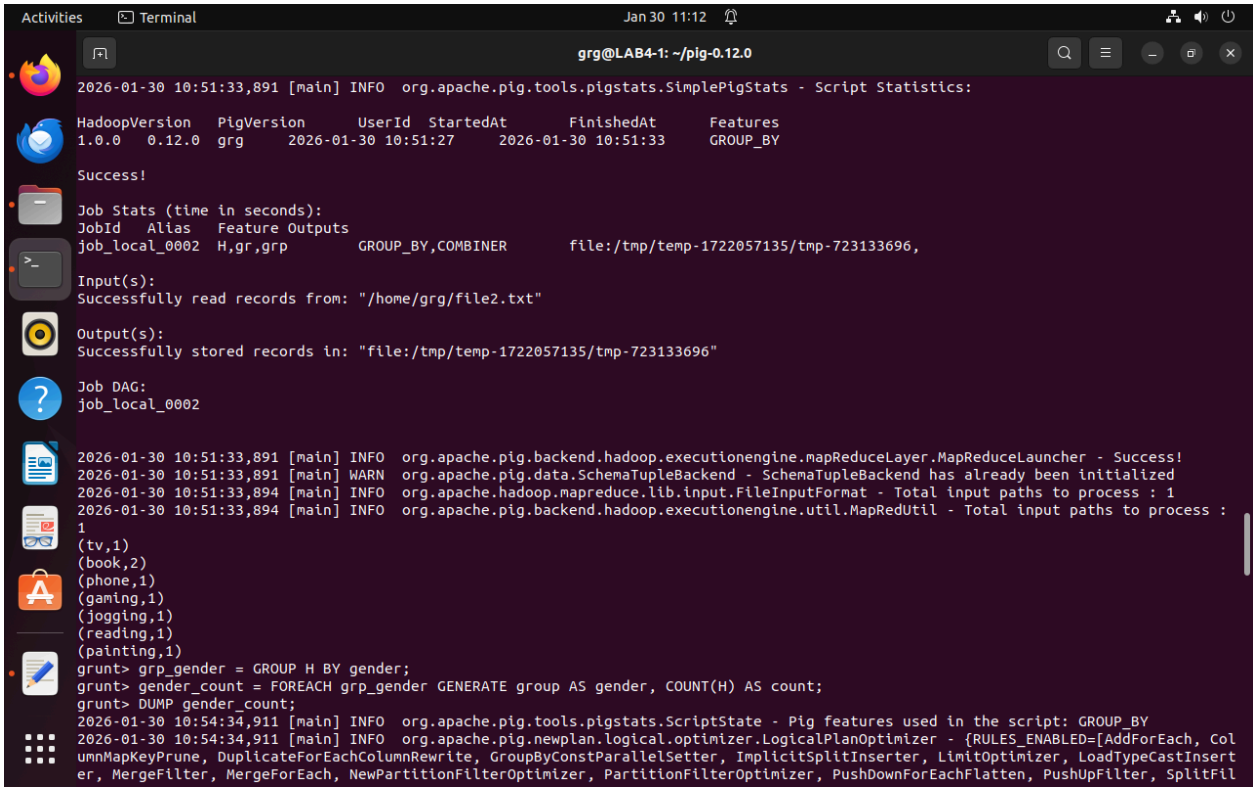
1,alice,female,jogging
2,anu,male,book
3,kalai,female,phone
4,mani,male,tv
5,laksh,female,book
6,keerthana,female,painting
7,vikas,male,reading
8,rohit,male,gaming

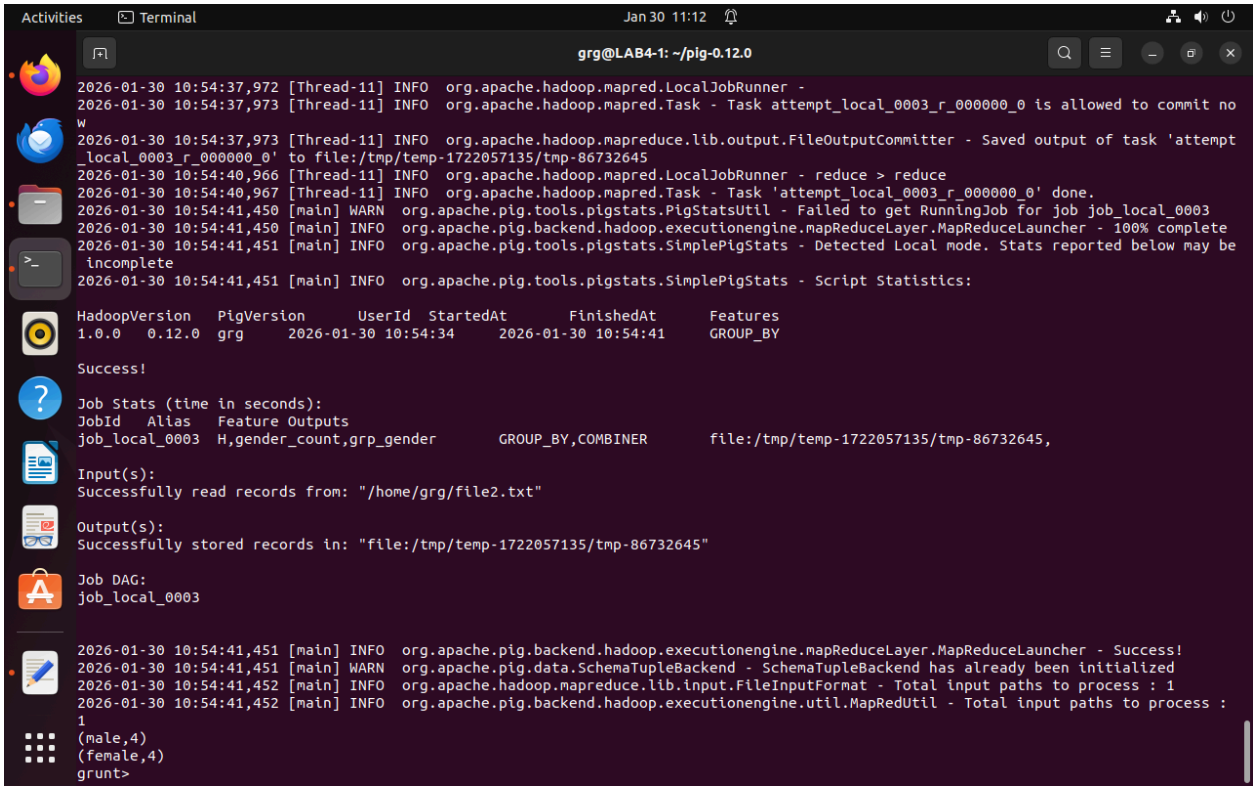
Code

```
H= LOAD' /home/grg/file2.txt' using PigStorage(',') as
(id:int,name:chararray,gender:chararray,hobby:chararray);

grp = group H by hobby;

gr = FOREACH grp GENERATE group,COUNT(H);

dump gr;

grp_gender = GROUP H BY gender;

gender_count = FOREACH grp_gender GENERATE group AS gender, COUNT(H) AS count;

DUMP gender_count;
```

Output:


Activities
Terminal
Jan 30 11:11
grg@LAB4-1: ~/pig-0.12.0
grunt> H= LOAD' /home/grg/file2.txt' using PigStorage(',') as (id:int,name:chararray,gender:chararray,hobby:chararray);
grunt> grp = group H by hobby;
grunt> gr = FOREACH grp GENERATE group,COUNT(H);
grunt> dump gr;
2026-01-30 10:51:27,359 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: GROUP_BY
2026-01-30 10:51:27,359 [main] INFO org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, DuplicateForEachColumnRewrite, GroupByConstParallelSetter, ImplicitSplitInserter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NewPartitionFilterOptimizer, PartitionFilterOptimizer, PushDownForEachFlatten, PushUpFilter, SplitFilter, StreamTypeCastInserter], RULES_DISABLED=[FilterLogicExpressionSimplifier]}
2026-01-30 10:51:27,361 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2026-01-30 10:51:27,362 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.CombinerOptimizer - Choosing to move algebraic foreach to combiner
2026-01-30 10:51:27,362 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2026-01-30 10:51:27,362 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
2026-01-30 10:51:27,363 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig script settings are added to the job
2026-01-30 10:51:27,363 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffer.percent is not set, set to default 0.3
2026-01-30 10:51:27,364 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Setting up single store job
2026-01-30 10:51:27,365 [main] INFO org.apache.pig.data.SchemaTupleFrontend - Key [pig.schematuple] is false, will not generate code.
2026-01-30 10:51:27,365 [main] INFO org.apache.pig.data.SchemaTupleFrontend - Starting process to move generated code to distributed cache
2026-01-30 10:51:27,365 [main] INFO org.apache.pig.data.SchemaTupleFrontend - Distributed cache not supported or needed in local mode. Setting key [pig.schematuple.local.dir] with code temp directory: /tmp/1769750487365-0
2026-01-30 10:51:27,365 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Reduce phase detected, estimating # of required reducers.
2026-01-30 10:51:27,365 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Using reducer estimator: org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.InputSizeReducerEstimator
2026-01-30 10:51:27,365 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.InputSizeReducerEstimator - BytesPerReducer=1000000000 maxReducers=999 totalInputFileSize=-1
2026-01-30 10:51:27,365 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Could not estimate number of reducers and no requested or default parallelism set. Defaulting to 1 reducer.
2026-01-30 10:51:27,365 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Setting Parallelism to 1
2026-01-30 10:51:27,370 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 1 map-reduce job(s) waiting for submission.


Activities
Terminal
Jan 30 11:12
grg@LAB4-1: ~/pig-0.12.0
2026-01-30 10:51:33,891 [main] INFO org.apache.pig.tools.pigstats.SimplePigStats - Script Statistics:
HadoopVersion PigVersion UserId StartedAt FinishedAt Features
1.0.0 0.12.0 grg 2026-01-30 10:51:27 2026-01-30 10:51:33 GROUP_BY
Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs
job_local_0002 H,gr,grp GROUP_BY,COMBINER file:/tmp/temp-1722057135/tmp-723133696,
Input(s):
Successfully read records from: "/home/grg/file2.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp-1722057135/tmp-723133696"
Job DAG:
job_local_0002
2026-01-30 10:51:33,891 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2026-01-30 10:51:33,891 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2026-01-30 10:51:33,894 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2026-01-30 10:51:33,894 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(tv,1)
(book,2)
(phone,1)
(gaming,1)
(jogging,1)
(reading,1)
(painting,1)
grunt> grp_gender = GROUP H BY gender;
grunt> gender_count = FOREACH grp_gender GENERATE group AS gender, COUNT(H) AS count;
grunt> DUMP gender_count;
2026-01-30 10:54:34,911 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: GROUP_BY
2026-01-30 10:54:34,911 [main] INFO org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, DuplicateForEachColumnRewrite, GroupByConstParallelSetter, ImplicitSplitInserter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NewPartitionFilterOptimizer, PartitionFilterOptimizer, PushDownForEachFlatten, PushUpFilter, SplitFil



5) Pig script -Numbers of words

Check in file home page **input.txt**
Or
No input file means
Create new file in texteditor—**input.txt**

Apache Pig is a high-level data processing platform that runs on Hadoop.It is designed to handle large datasets efficiently using parallel processing.The scripting language used in Apache Pig is called Pig Latin.Pig Latin provides a simple, SQL-like syntax for writing data analysis programs. Apache Pig is commonly used for ETL tasks, data transformation, and analysis.Apache Pig acts as an abstraction layer over Hadoop's MapReduce framework.It enables developers to write fewer lines of code for complex data workflows.Pig Latin scripts are automatically optimized and converted into MapReduce jobs.

Coding

inputfile = LOAD '/home/grg/input.txt' AS (line:chararray);

DUMP inputfile;

words = FOREACH inputfile GENERATE FLATTEN(TOKENIZE(line)) AS word;

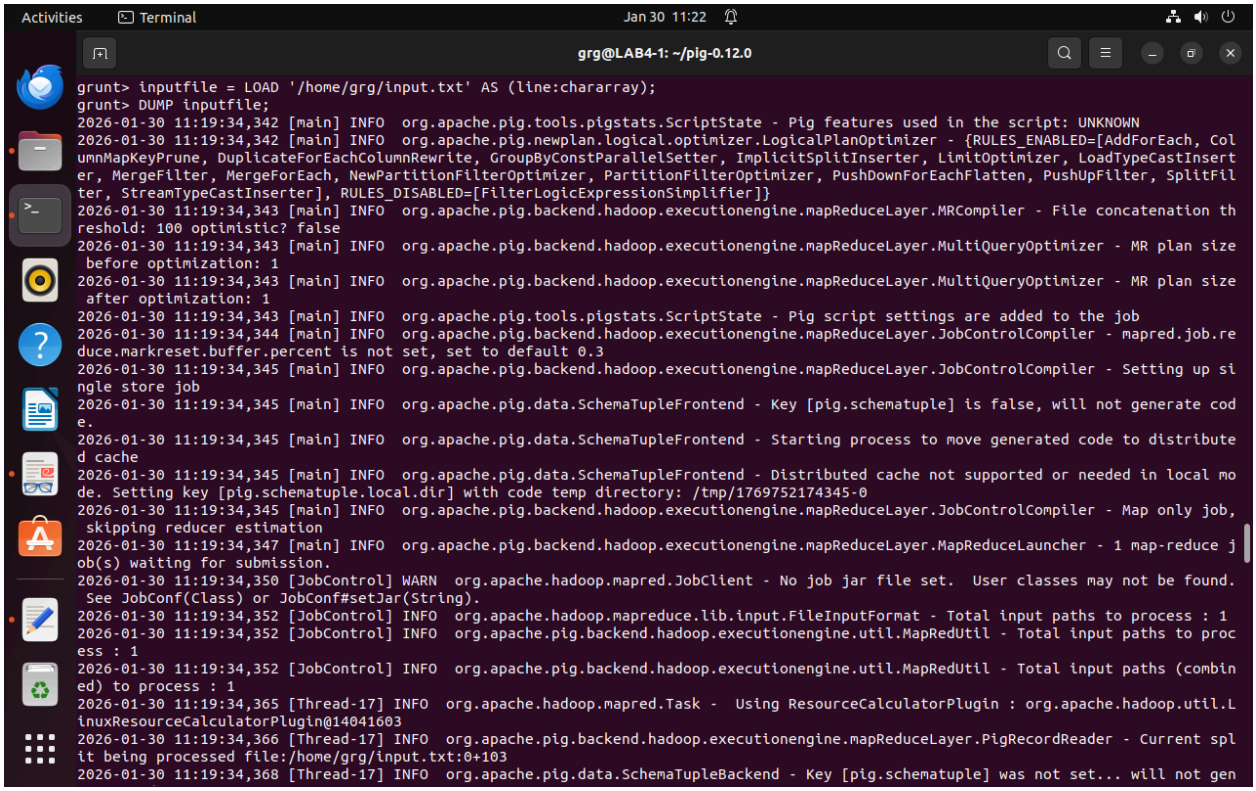
filtered_words = FILTER words BY word MATCHES '\\w+';

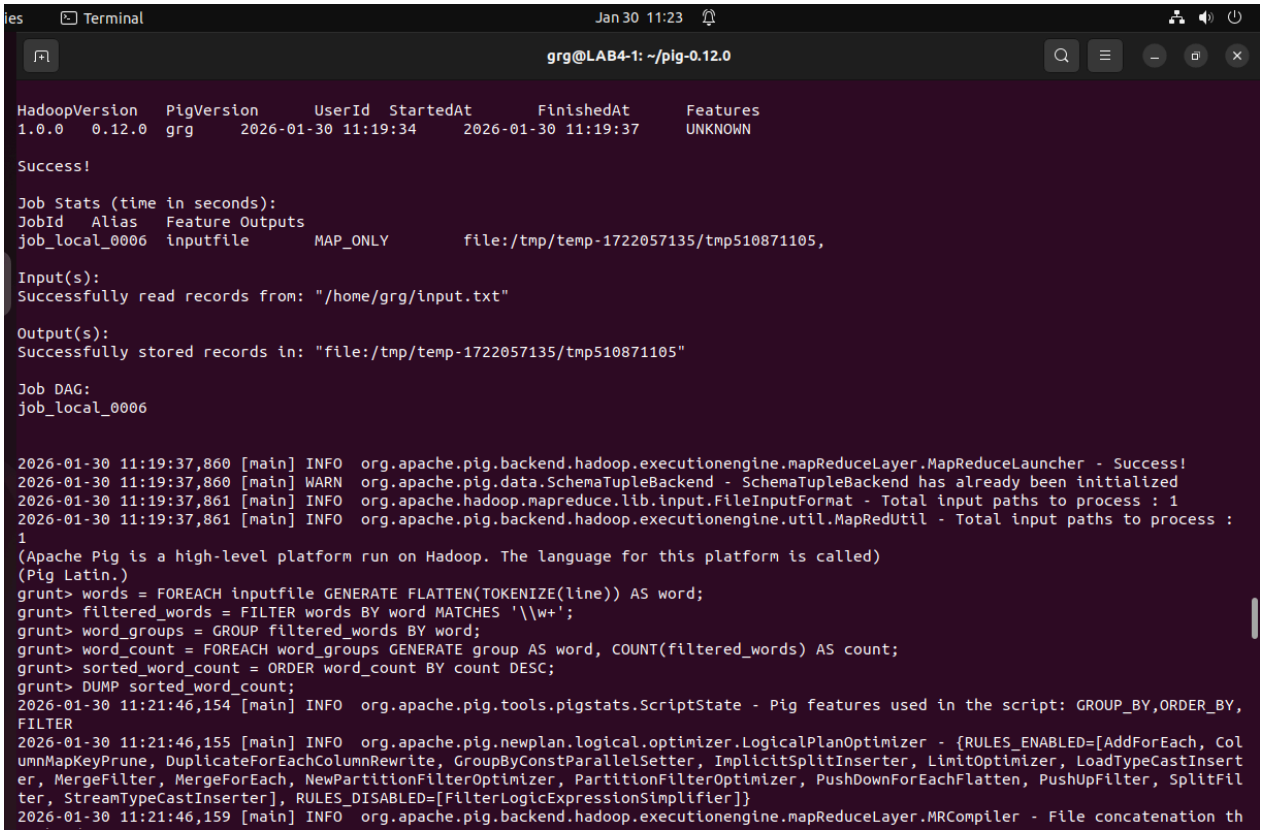
word_groups = GROUP filtered_words BY word;

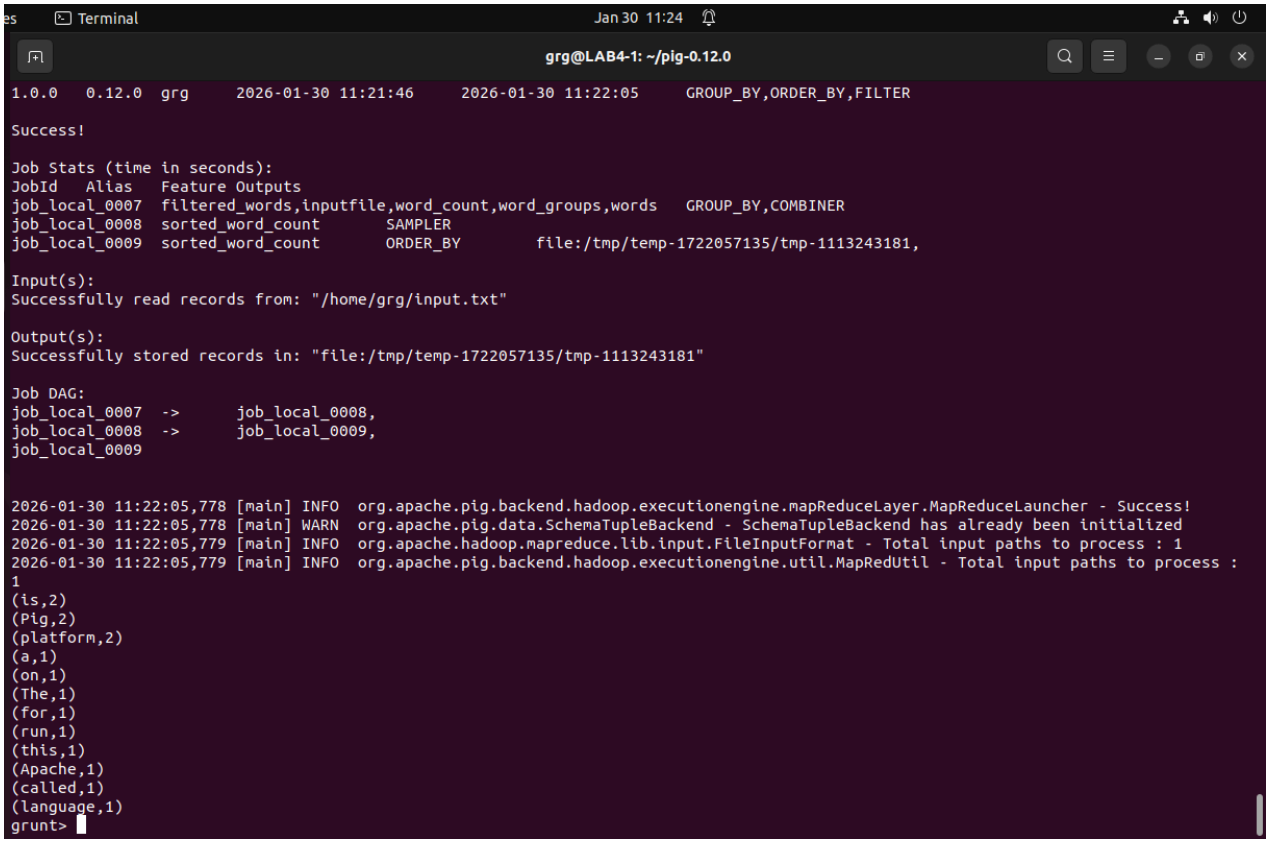
word_count = FOREACH word_groups GENERATE group AS word, COUNT(filtered_words) AS count;

sorted_word_count = ORDER word_count BY count DESC;

DUMP sorted_word_count;



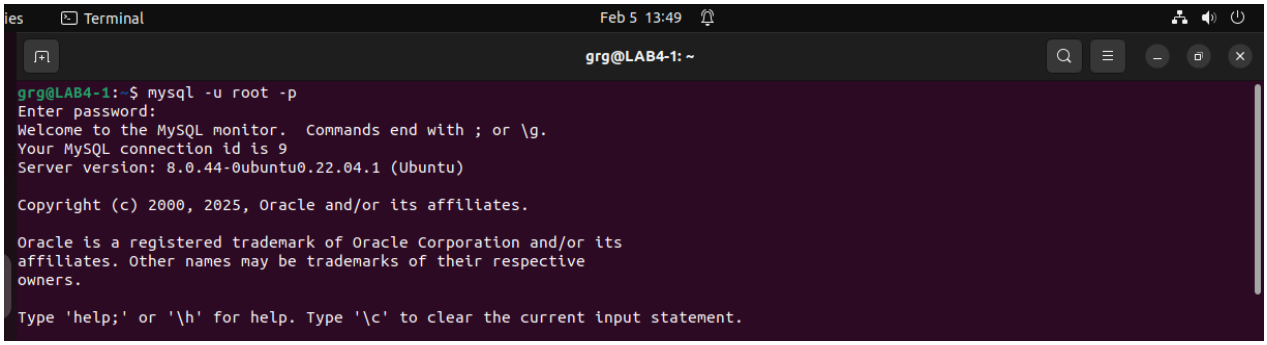

ies
Terminal
Jan 30 11:23
grg@LAB4-1: ~/pig-0.12.0
HadoopVersion PigVersion UserId StartedAt FinishedAt Features
1.0.0 0.12.0 grg 2026-01-30 11:19:34 2026-01-30 11:19:37 UNKNOWN
Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs
job_local_0006 inputfile MAP_ONLY file:/tmp/temp-1722057135/tmp510871105,
Input(s):
Successfully read records from: "/home/grg/input.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp-1722057135/tmp510871105"
Job DAG:
job_local_0006
2026-01-30 11:19:37,860 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2026-01-30 11:19:37,860 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2026-01-30 11:19:37,861 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2026-01-30 11:19:37,861 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process :
1
(Apache Pig is a high-level platform run on Hadoop. The language for this platform is called)
(Pig Latin.)
grunt> words = FOREACH inputfile GENERATE FLATTEN(TOKENIZE(line)) AS word;
grunt> filtered_words = FILTER words BY word MATCHES '\\w+';
grunt> word_groups = GROUP filtered_words BY word;
grunt> word_count = FOREACH word_groups GENERATE group AS word, COUNT(filtered_words) AS count;
grunt> sorted_word_count = ORDER word_count BY count DESC;
grunt> DUMP sorted_word_count;
2026-01-30 11:21:46,154 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: GROUP_BY,ORDER_BY,
FILTER
2026-01-30 11:21:46,155 [main] INFO org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, Col
umnMapKeyPrune, DuplicateForEachColumnRewrite, GroupByConstParallelSetter, ImplicitSplitInserter, LimitOptimizer, LoadTypeCastInsert
er, MergeFilter, MergeForEach, NewPartitionFilterOptimizer, PartitionFilterOptimizer, PushDownForEachFlatten, PushUpFilter, SplitFil
ter, StreamTypeCastInserter], RULES_DISABLED=[FilterLogicExpressionSimplifier]}
2026-01-30 11:21:46,159 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation th


es
Terminal
Jan 30 11:24
grg@LAB4-1: ~/pig-0.12.0
1.0.0 0.12.0 grg 2026-01-30 11:21:46 2026-01-30 11:22:05 GROUP_BY,ORDER_BY,FILTER
Success!
Job Stats (time in seconds):
JobId Alias Feature Outputs
job_local_0007 filtered_words,inputfile,word_count,word_groups,words GROUP_BY,COMBINER
job_local_0008 sorted_word_count SAMPLER
job_local_0009 sorted_word_count ORDER_BY file:/tmp/temp-1722057135/tmp-1113243181,
Input(s):
Successfully read records from: "/home/grg/input.txt"
Output(s):
Successfully stored records in: "file:/tmp/temp-1722057135/tmp-1113243181"
Job DAG:
job_local_0007 -> job_local_0008,
job_local_0008 -> job_local_0009,
job_local_0009
2026-01-30 11:22:05,778 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2026-01-30 11:22:05,778 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2026-01-30 11:22:05,779 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2026-01-30 11:22:05,779 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process :
1
(is,2)
(Pig,2)
(platform,2)
(a,1)
(on,1)
(The,1)
(for,1)
(run,1)
(this,1)
(Apache,1)
(called,1)
(language,1)
grunt>

Hive setup

Open terminal

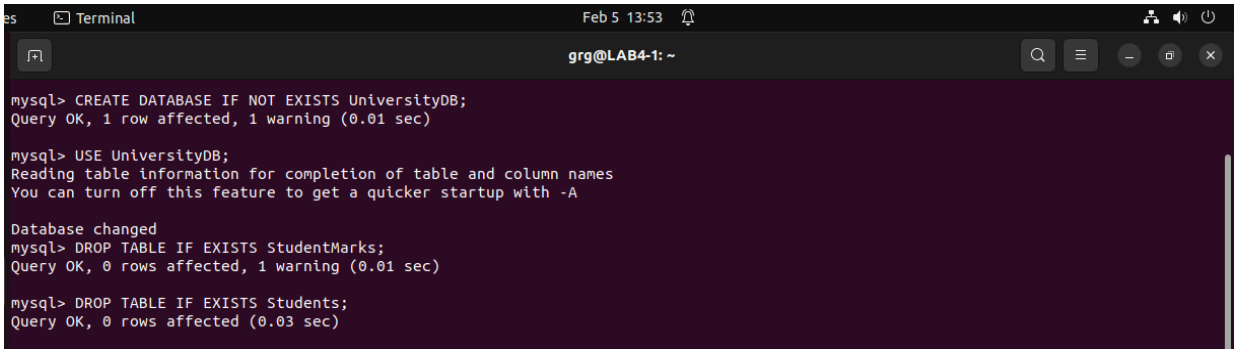
mysql -u root -p
Password : root

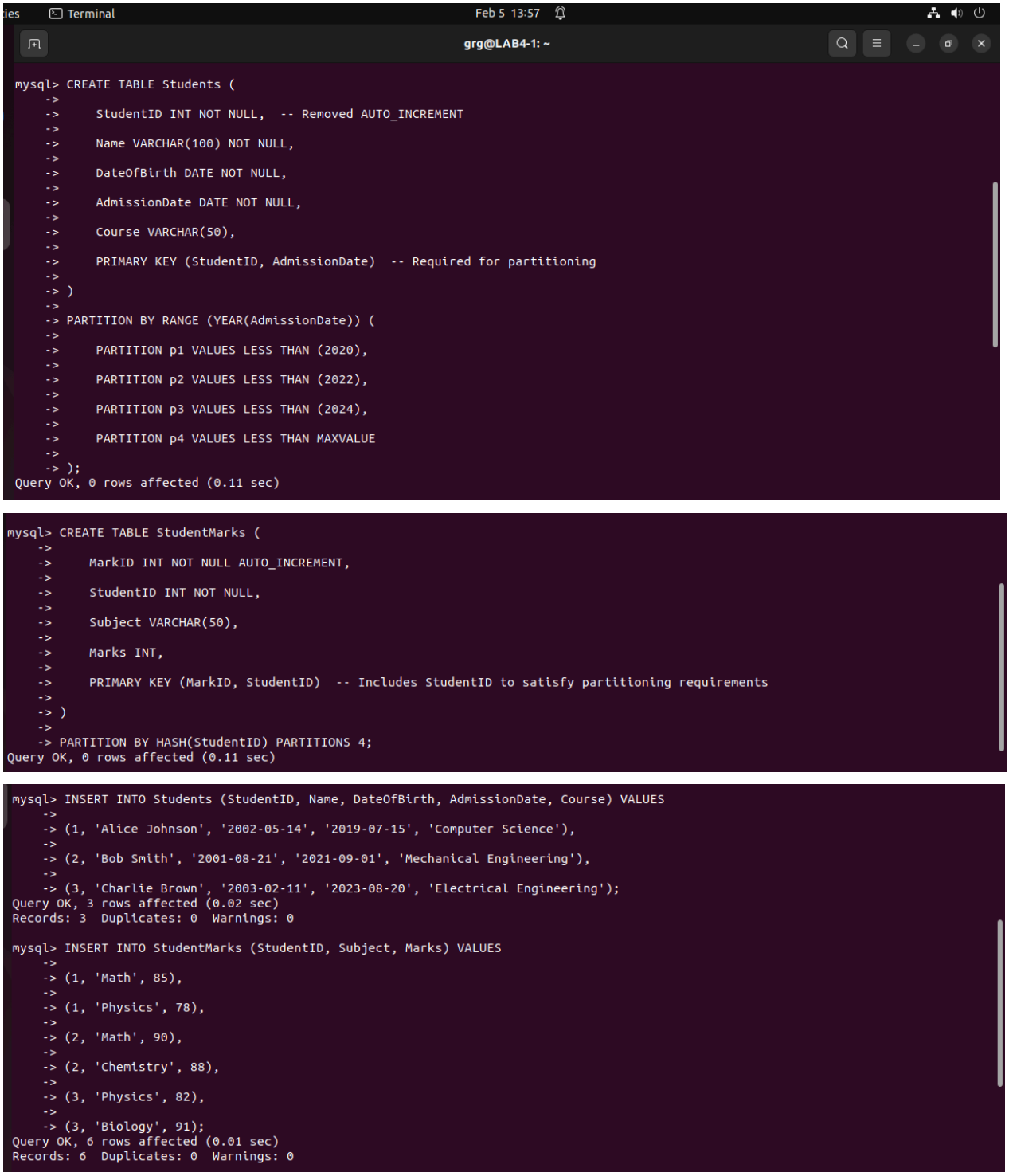


7) Implements Tables,partitions,and buckets in hive

```
CREATE DATABASE IF NOT EXISTS UniversityDB;

USE UniversityDB;

DROP TABLE IF EXISTS StudentMarks;

DROP TABLE IF EXISTS Students;

CREATE TABLE Students (
StudentID INT NOT NULL,  -- Removed AUTO_INCREMENT
Name VARCHAR(100) NOT NULL,
DateOfBirth DATE NOT NULL,
AdmissionDate DATE NOT NULL,
Course VARCHAR(50),
PRIMARY KEY (StudentID, AdmissionDate)  -- Required for partitioning
)
PARTITION BY RANGE (YEAR(AdmissionDate)) (
PARTITION p1 VALUES LESS THAN (2020),
PARTITION p2 VALUES LESS THAN (2022),
PARTITION p3 VALUES LESS THAN (2024),
PARTITION p4 VALUES LESS THAN MAXVALUE
);
```

```
CREATE TABLE StudentMarks (
MarkID INT NOT NULL AUTO_INCREMENT,
StudentID INT NOT NULL,
Subject VARCHAR(50),
Marks INT,
PRIMARY KEY (MarkID, StudentID)  -- Includes StudentID to satisfy partitioning requirements
)
PARTITION BY HASH(StudentID) PARTITIONS 4;

INSERT INTO Students (StudentID, Name, DateOfBirth, AdmissionDate, Course) VALUES
(1, 'Alice Johnson', '2002-05-14', '2019-07-15', 'Computer Science'),
(2, 'Bob Smith', '2001-08-21', '2021-09-01', 'Mechanical Engineering'),
(3, 'Charlie Brown', '2003-02-11', '2023-08-20', 'Electrical Engineering');

INSERT INTO StudentMarks (StudentID, Subject, Marks) VALUES
(1, 'Math', 85),
(1, 'Physics', 78),
(2, 'Math', 90),
(2, 'Chemistry', 88),
 (3, 'Physics', 82),
(3, 'Biology', 91);

SELECT * FROM Students;

SELECT * FROM StudentMarks;

SELECT PARTITION_NAME, TABLE_NAME, TABLE_SCHEMA
FROM INFORMATION_SCHEMA.PARTITIONS
WHERE TABLE_NAME = 'StudentMarks';
```






```
mysql> CREATE TABLE Students (
    ->
    ->     StudentID INT NOT NULL,  -- Removed AUTO_INCREMENT
    ->
    ->     Name VARCHAR(100) NOT NULL,
    ->
    ->     DateOfBirth DATE NOT NULL,
    ->
    ->     AdmissionDate DATE NOT NULL,
    ->
    ->     Course VARCHAR(50),
    ->
    ->     PRIMARY KEY (StudentID, AdmissionDate)  -- Required for partitioning
    ->
    -> )
    ->
    -> PARTITION BY RANGE (YEAR(AdmissionDate)) (
    ->
    ->     PARTITION p1 VALUES LESS THAN (2020),
    ->
    ->     PARTITION p2 VALUES LESS THAN (2022),
    ->
    ->     PARTITION p3 VALUES LESS THAN (2024),
    ->
    ->     PARTITION p4 VALUES LESS THAN MAXVALUE
    ->
    -> );
Query OK, 0 rows affected (0.11 sec)
```

```
mysql> CREATE TABLE StudentMarks (
    ->
    ->     MarkID INT NOT NULL AUTO_INCREMENT,
    ->
    ->     StudentID INT NOT NULL,
    ->
    ->     Subject VARCHAR(50),
    ->
    ->     Marks INT,
    ->
    ->     PRIMARY KEY (MarkID, StudentID)  -- Includes StudentID to satisfy partitioning requirements
    ->
    -> )
    ->
    -> PARTITION BY HASH(StudentID) PARTITIONS 4;
Query OK, 0 rows affected (0.11 sec)
```

```
mysql> INSERT INTO Students (StudentID, Name, DateOfBirth, AdmissionDate, Course) VALUES
    ->
    -> (1, 'Alice Johnson', '2002-05-14', '2019-07-15', 'Computer Science'),
    ->
    -> (2, 'Bob Smith', '2001-08-21', '2021-09-01', 'Mechanical Engineering'),
    ->
    -> (3, 'Charlie Brown', '2003-02-11', '2023-08-20', 'Electrical Engineering');
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO StudentMarks (StudentID, Subject, Marks) VALUES
    ->
    -> (1, 'Math', 85),
    ->
    -> (1, 'Physics', 78),
    ->
    -> (2, 'Math', 90),
    ->
    -> (2, 'Chemistry', 88),
    ->
    -> (3, 'Physics', 82),
    ->
    -> (3, 'Biology', 91);
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT * FROM Students;
+-----------+---------------+-------------+--------------+-----------------------+
| StudentID | Name          | DateOfBirth | AdmissionDate | Course                |
+-----------+---------------+-------------+--------------+-----------------------+
|         1 | Alice Johnson | 2002-05-14  | 2019-07-15    | Computer Science      |
|         2 | Bob Smith     | 2001-08-21  | 2021-09-01    | Mechanical Engineering |
|         3 | Charlie Brown | 2003-02-11  | 2023-08-20    | Electrical Engineering |
+-----------+---------------+-------------+--------------+-----------------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM StudentMarks;
+--------+-----------+-----------+-------+
| MarkID | StudentID | Subject   | Marks |
+--------+-----------+-----------+-------+
|      1 |         1 | Math      |    85 |
|      2 |         1 | Physics   |    78 |
|      3 |         2 | Math      |    90 |
|      4 |         2 | Chemistry |    88 |
|      5 |         3 | Physics   |    82 |
|      6 |         3 | Biology   |    91 |
+--------+-----------+-----------+-------+
6 rows in set (0.00 sec)
```

```
mysql> SELECT PARTITION_NAME, TABLE_NAME, TABLE_SCHEMA
    ->
    -> FROM INFORMATION_SCHEMA.PARTITIONS
    ->
    -> WHERE TABLE_NAME = 'StudentMarks';
+----------------+-------------+--------------+
| PARTITION_NAME | TABLE_NAME  | TABLE_SCHEMA |
+----------------+-------------+--------------+
| p3             | StudentMarks | UniversityDB |
| p2             | StudentMarks | UniversityDB |
| p1             | StudentMarks | UniversityDB |
| p0             | StudentMarks | UniversityDB |
+----------------+-------------+--------------+
4 rows in set (0.00 sec)

mysql>
```

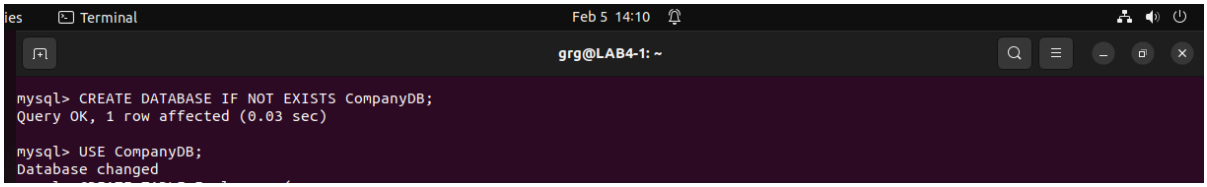
8) Built In Functions in Hive

CREATE DATABASE IF NOT EXISTS CompanyDB;
USE CompanyDB;

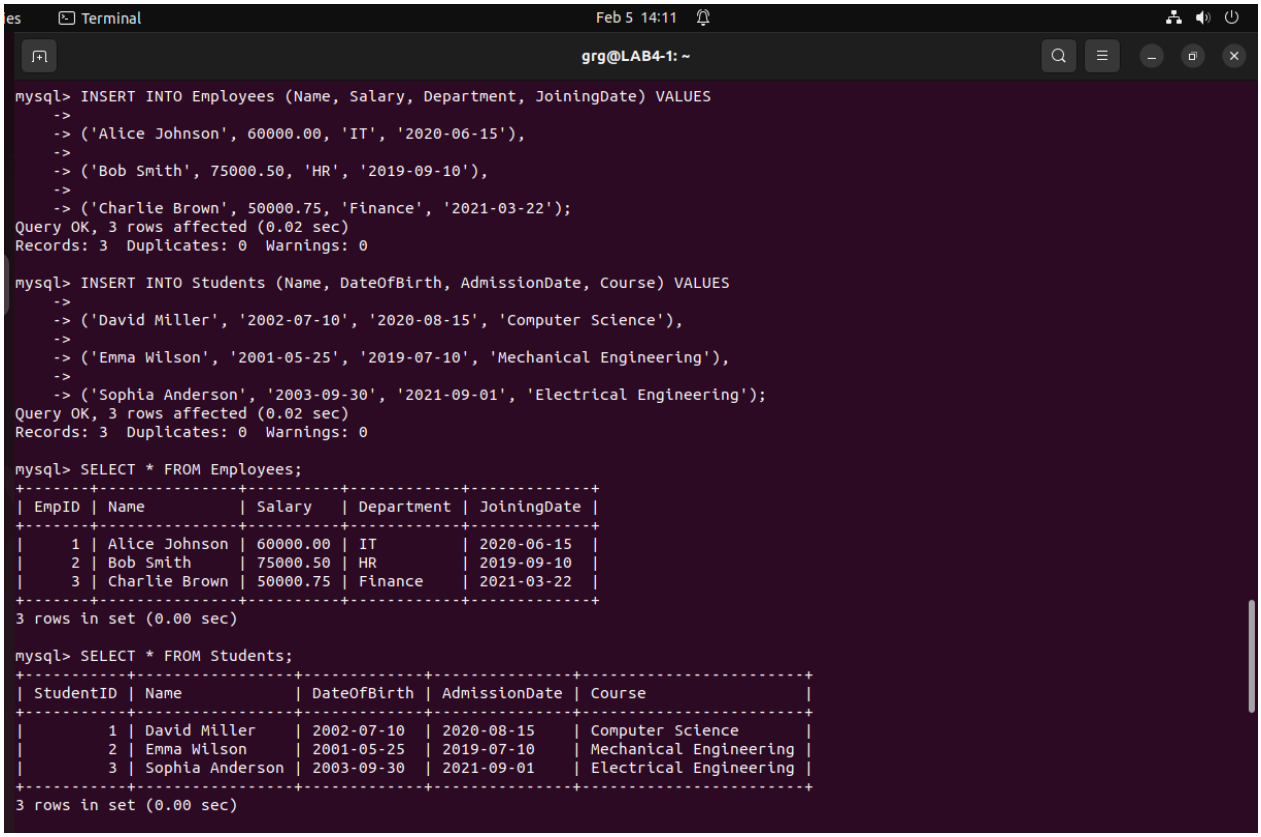
CREATE TABLE Employees (
EmpID INT AUTO_INCREMENT PRIMARY KEY,
Name VARCHAR(100),
Salary DECIMAL(10,2),
Department VARCHAR(50),
JoiningDate DATE
);

CREATE TABLE Students (
StudentID INT AUTO_INCREMENT PRIMARY KEY,
Name VARCHAR(100),
DateOfBirth DATE,
AdmissionDate DATE,
Course VARCHAR(50)
);

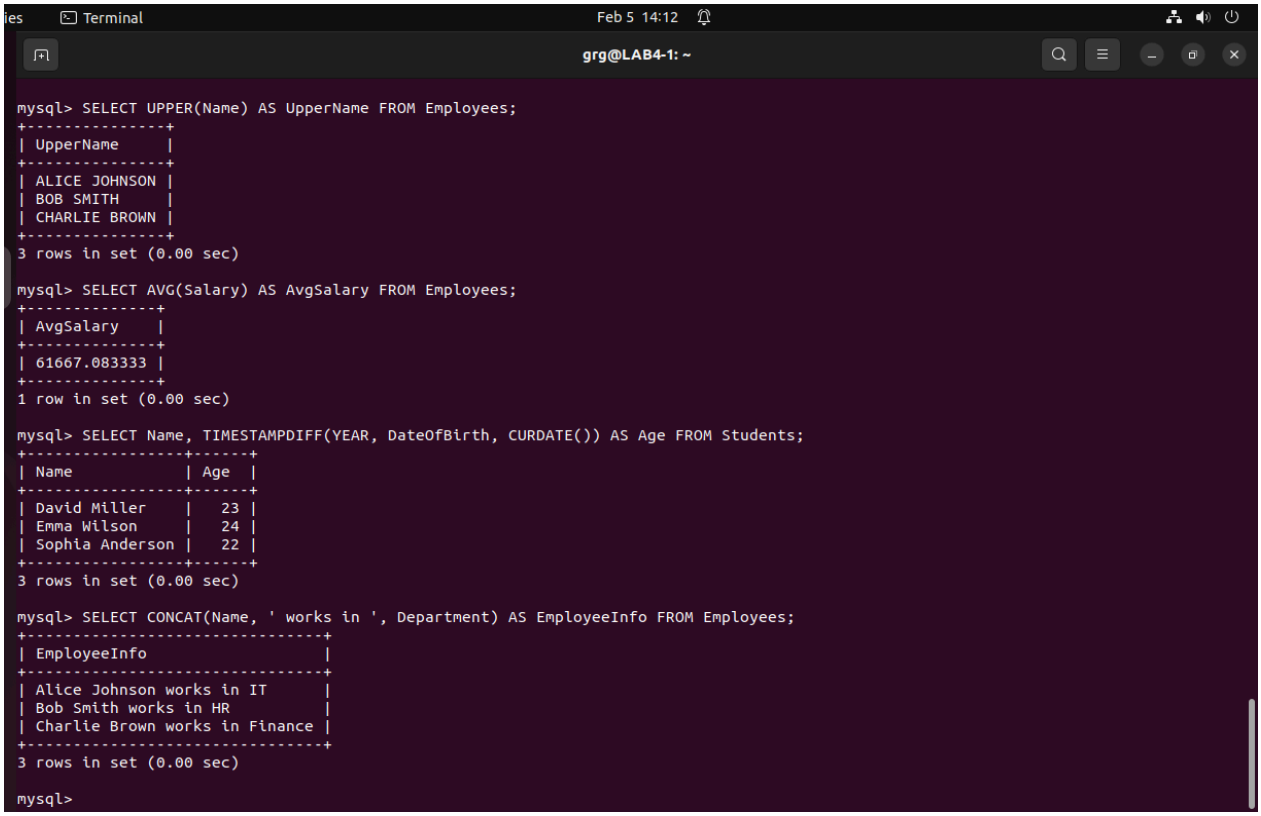
```
INSERT INTO Employees (Name, Salary, Department, JoiningDate) VALUES
('Alice Johnson', 60000.00, 'IT', '2020-06-15'),
('Bob Smith', 75000.50, 'HR', '2019-09-10'),
('Charlie Brown', 50000.75, 'Finance', '2021-03-22');

INSERT INTO Students (Name, DateOfBirth, AdmissionDate, Course) VALUES
('David Miller', '2002-07-10', '2020-08-15', 'Computer Science'),
('Emma Wilson', '2001-05-25', '2019-07-10', 'Mechanical Engineering'),
('Sophia Anderson', '2003-09-30', '2021-09-01', 'Electrical Engineering');

SELECT * FROM Employees;

SELECT * FROM Students;

SELECT UPPER(Name) AS UpperName FROM Employees;

SELECT AVG(Salary) AS AvgSalary FROM Employees;

SELECT Name, TIMESTAMPDIFF(YEAR, DateOfBirth, CURDATE()) AS Age FROM
Students;

SELECT CONCAT(Name, ' works in ', Department) AS EmployeeInfo FROM Employees;
```



```
mysql> CREATE TABLE Employees (
    ->
    ->     EmpID INT AUTO_INCREMENT PRIMARY KEY,
    ->
    ->     Name VARCHAR(100),
    ->
    ->     Salary DECIMAL(10,2),
    ->
    ->     Department VARCHAR(50),
    ->
    ->     JoiningDate DATE
    ->
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE Students (
    ->
    ->     StudentID INT AUTO_INCREMENT PRIMARY KEY,
    ->
    ->     Name VARCHAR(100),
    ->
    ->     DateOfBirth DATE,
    ->
    ->     AdmissionDate DATE,
    ->
    ->     Course VARCHAR(50)
    ->
    -> );
Query OK, 0 rows affected (0.05 sec)
```




```
mysql> INSERT INTO Employees (Name, Salary, Department, JoiningDate) VALUES
    ->
    -> ('Alice Johnson', 60000.00, 'IT', '2020-06-15'),
    ->
    -> ('Bob Smith', 75000.50, 'HR', '2019-09-10'),
    ->
    -> ('Charlie Brown', 50000.75, 'Finance', '2021-03-22');
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Students (Name, DateOfBirth, AdmissionDate, Course) VALUES
    ->
    -> ('David Miller', '2002-07-10', '2020-08-15', 'Computer Science'),
    ->
    -> ('Emma Wilson', '2001-05-25', '2019-07-10', 'Mechanical Engineering'),
    ->
    -> ('Sophia Anderson', '2003-09-30', '2021-09-01', 'Electrical Engineering');
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Employees;
+-------+---------------+----------+------------+-------------+
| EmpID | Name          | Salary   | Department | JoiningDate |
+-------+---------------+----------+------------+-------------+
|     1 | Alice Johnson | 60000.00 | IT         | 2020-06-15  |
|     2 | Bob Smith     | 75000.50 | HR         | 2019-09-10  |
|     3 | Charlie Brown | 50000.75 | Finance    | 2021-03-22  |
+-------+---------------+----------+------------+-------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM Students;
+-----------+-----------------+-------------+---------------+-----------------------+
| StudentID | Name            | DateOfBirth | AdmissionDate | Course                |
+-----------+-----------------+-------------+---------------+-----------------------+
|         1 | David Miller    | 2002-07-10  | 2020-08-15    | Computer Science      |
|         2 | Emma Wilson     | 2001-05-25  | 2019-07-10    | Mechanical Engineering |
|         3 | Sophia Anderson | 2003-09-30  | 2021-09-01    | Electrical Engineering |
+-----------+-----------------+-------------+---------------+-----------------------+
3 rows in set (0.00 sec)
```



9) Hive in Join operation in customer table

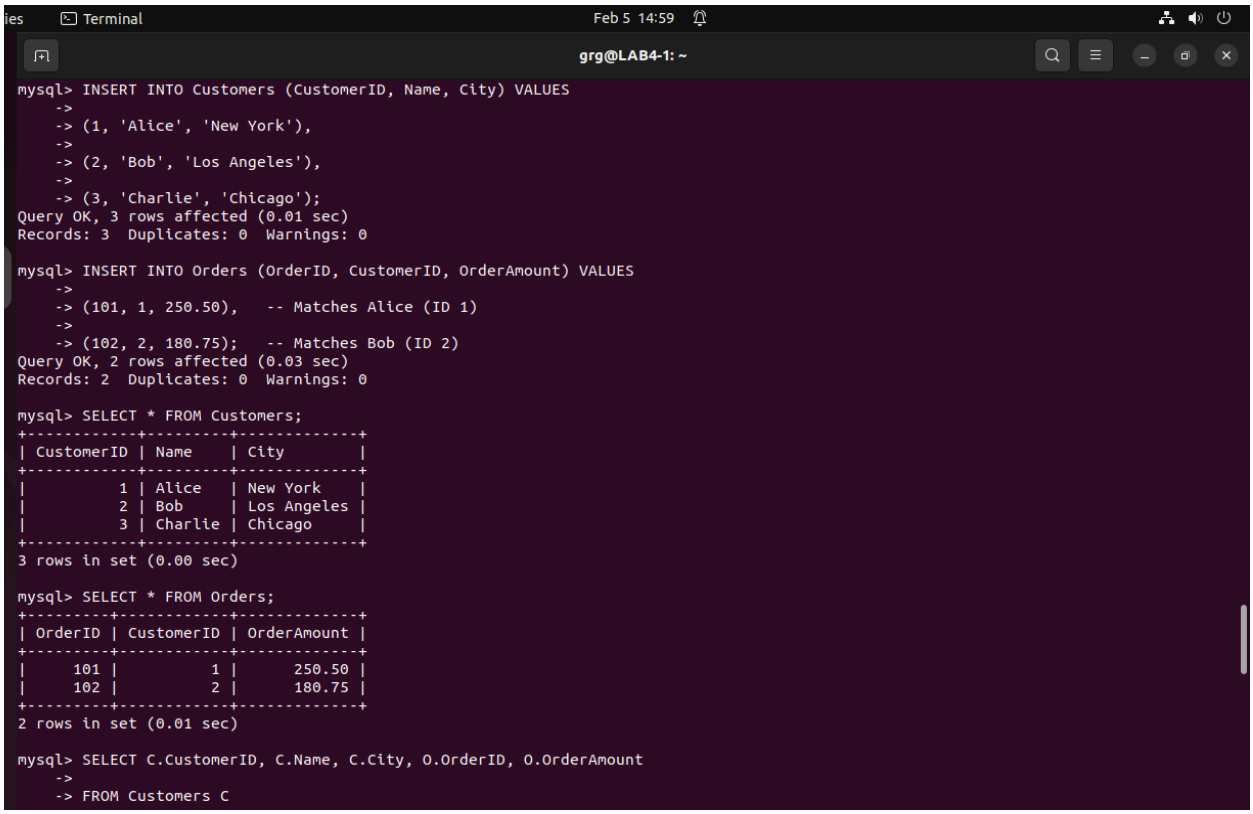
CREATE TABLE Customers (
CustomerID INT PRIMARY KEY,
Name VARCHAR(100),
City VARCHAR(50)
);

CREATE TABLE Orders (
OrderID INT PRIMARY KEY,
CustomerID INT,
OrderAmount DECIMAL(10,2),
FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

INSERT INTO Customers (CustomerID, Name, City) VALUES
(1, 'Alice', 'New York'),
(2, 'Bob', 'Los Angeles'),
(3, 'Charlie', 'Chicago');

```
INSERT INTO Orders (OrderID, CustomerID, OrderAmount) VALUES
(101, 1, 250.50),   -- Matches Alice (ID 1)
(102, 2, 180.75);   -- Matches Bob (ID 2)

SELECT * FROM Customers;

SELECT * FROM Orders;

SELECT C.CustomerID, C.Name, C.City, O.OrderID, O.OrderAmount
FROM Customers C
INNER JOIN Orders O
ON C.CustomerID = O.CustomerID;

SELECT C.CustomerID, C.Name, C.City, O.OrderID, O.OrderAmount
FROM Customers C
LEFT JOIN Orders O
ON C.CustomerID = O.CustomerID;

SELECT C.CustomerID, C.Name, C.City, O.OrderID, O.OrderAmount
FROM Customers C
RIGHT JOIN Orders O
ON C.CustomerID = O.CustomerID;

SELECT C.CustomerID, C.Name, C.City, O.OrderID, O.OrderAmount
FROM Customers C
LEFT JOIN Orders O
ON C.CustomerID = O.CustomerID
UNION
SELECT C.CustomerID, C.Name, C.City, O.OrderID, O.OrderAmount
FROM Customers C
RIGHT JOIN Orders O
ON C.CustomerID = O.CustomerID;
```


```
mysql> CREATE TABLE Customers (
    ->
    ->     CustomerID INT PRIMARY KEY,
    ->
    ->     Name VARCHAR(100),
    ->
    ->     City VARCHAR(50)
    ->
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE Orders (
    ->
    ->     OrderID INT PRIMARY KEY,
    ->
    ->     CustomerID INT,
    ->
    ->     OrderAmount DECIMAL(10,2),
    ->
    ->     FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
    ->
    -> );
Query OK, 0 rows affected (0.04 sec)
```




```
mysql> INSERT INTO Customers (CustomerID, Name, City) VALUES
    ->
    -> (1, 'Alice', 'New York'),
    ->
    -> (2, 'Bob', 'Los Angeles'),
    ->
    -> (3, 'Charlie', 'Chicago');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Orders (OrderID, CustomerID, OrderAmount) VALUES
    ->
    -> (101, 1, 250.50),   -- Matches Alice (ID 1)
    ->
    -> (102, 2, 180.75);   -- Matches Bob (ID 2)
Query OK, 2 rows affected (0.03 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Customers;
+------------+---------+-------------+
| CustomerID | Name    | City        |
+------------+---------+-------------+
|          1 | Alice   | New York    |
|          2 | Bob     | Los Angeles |
|          3 | Charlie | Chicago     |
+------------+---------+-------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM Orders;
+---------+------------+-------------+
| OrderID | CustomerID | OrderAmount |
+---------+------------+-------------+
|     101 |          1 |      250.50 |
|     102 |          2 |      180.75 |
+---------+------------+-------------+
2 rows in set (0.01 sec)

mysql> SELECT C.CustomerID, C.Name, C.City, O.OrderID, O.OrderAmount
    ->
    -> FROM Customers C
```

```
mysql> SELECT C.CustomerID, C.Name, C.City, O.OrderID, O.OrderAmount
    ->
    -> FROM Customers C
    ->
    -> INNER JOIN Orders O
    ->
    -> ON C.CustomerID = O.CustomerID;
+------------+-------+-------------+---------+-------------+
| CustomerID | Name  | City        | OrderID | OrderAmount |
+------------+-------+-------------+---------+-------------+
|          1 | Alice | New York    |     101 |      250.50 |
|          2 | Bob   | Los Angeles |     102 |      180.75 |
+------------+-------+-------------+---------+-------------+
2 rows in set (0.00 sec)

mysql> SELECT C.CustomerID, C.Name, C.City, O.OrderID, O.OrderAmount
    ->
    -> FROM Customers C
    ->
    -> LEFT JOIN Orders O
    ->
    -> ON C.CustomerID = O.CustomerID;
+------------+---------+-------------+---------+-------------+
| CustomerID | Name    | City        | OrderID | OrderAmount |
+------------+---------+-------------+---------+-------------+
|          1 | Alice   | New York    |     101 |      250.50 |
|          2 | Bob     | Los Angeles |     102 |      180.75 |
|          3 | Charlie | Chicago     |    NULL |        NULL |
+------------+---------+-------------+---------+-------------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT C.CustomerID, C.Name, C.City, O.OrderID, O.OrderAmount
    ->
    -> FROM Customers C
    ->
    -> RIGHT JOIN Orders O
    ->
    -> ON C.CustomerID = O.CustomerID;
+------------+-------+-------------+---------+-------------+
| CustomerID | Name  | City        | OrderID | OrderAmount |
+------------+-------+-------------+---------+-------------+
|          1 | Alice | New York    |     101 |      250.50 |
|          2 | Bob   | Los Angeles |     102 |      180.75 |
+------------+-------+-------------+---------+-------------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT C.CustomerID, C.Name, C.City, O.OrderID, O.OrderAmount
    ->
    -> FROM Customers C
    ->
    -> LEFT JOIN Orders O
    ->
    -> ON C.CustomerID = O.CustomerID
    ->
    -> UNION
    ->
    -> SELECT C.CustomerID, C.Name, C.City, O.OrderID, O.OrderAmount
    ->
    -> FROM Customers C
    ->
    -> RIGHT JOIN Orders O
    ->
    -> ON C.CustomerID = O.CustomerID;
+------------+---------+-------------+---------+-------------+
| CustomerID | Name    | City        | OrderID | OrderAmount |
+------------+---------+-------------+---------+-------------+
|          1 | Alice   | New York    |     101 |      250.50 |
|          2 | Bob     | Los Angeles |     102 |      180.75 |
|          3 | Charlie | Chicago     |    NULL |        NULL |
+------------+---------+-------------+---------+-------------+
3 rows in set (0.02 sec)
```

Map reduce

CHECK 6 DAEMONS IN TERMINAL AFTER CONTINUE THE CODE IN TERMINAL

PROGRAM 2:Develop a Map Reduce to find the maximum electrical consumption in each year given electrical consumption for each month in each year

IN TERMINAL

touch eb.csv

nano mapper2.py

nano reducer2.py

cat eb.csv

cat eb.csv | python3 mapper2.py | python3 reducer2.py > output2.txt

cat output2.txt

INSERT INTO eb.csv

Year,Month,Consumption

2020,Jan,320

2020,Feb,290

2020,Mar,450

2020,Apr,380

2021,Jan,500

2021,Feb,450

2021,Mar,510

2021,Apr,480

INSERT INTO mapper2.py

```
#!/usr/bin/env python3
import sys
for line in sys.stdin:
    line = line.strip()
```

```
parts = line.split(",")

# Skip header
if parts[0] == "Year":
    continue
try:
    year = parts[0]
    consumption = int(parts[2])  # Convert consumption to integer
    print(f"{year}\t{consumption}")
except ValueError:
    continue  # Skip invalid lines
```

After type the code press F2

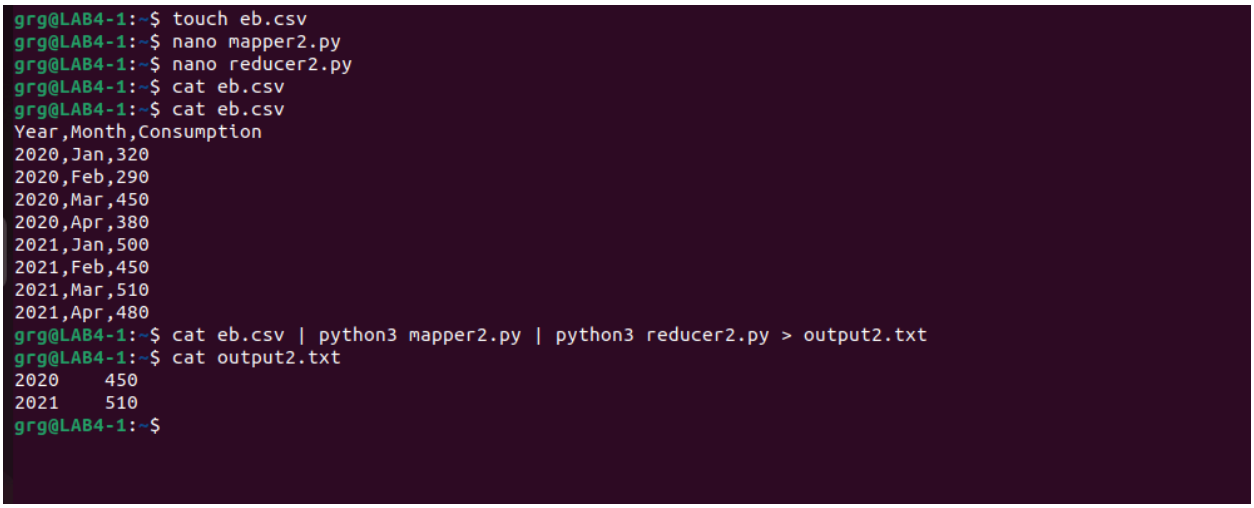
INSERT INTO reducer2.py

```
#!/usr/bin/env python3
import sys
current_year = None
max_consumption = 0
```

```
for line in sys.stdin:
    line = line.strip()
    year, consumption = line.split("\t")
    try:
        consumption = int(consumption)
        if current_year == year:
            max_consumption = max(max_consumption, consumption)
        else:
            if current_year:
                print(f"{current_year}\t{max_consumption}")  # Print max for previous year
            current_year = year
            max_consumption = consumption
    except ValueError:
        continue
# Print last year's max
if current_year:
    print(f"{current_year}\t{max_consumption}")
```

After type the code press F2

OUTPUT



—---

PROGRAM 3:Develop a Map Reduce to analyze weather data set and print whether the day is shiny or cool day.

TERMINAL

touch weather.csv

nano mapper1.py

nano reducer1.py

cat weather.csv

cat weather.csv | python3 mapper1.py | python3 reducer1.py > output.txt

cat output.txt

CREATE INTO weather.csv —-IN TEXT EDITOR

Date,Temperature,WeatherType

2024-01-01,35,Sunny

2024-01-02,18,Cloudy

2024-01-03,42,Sunny

2024-01-04,12,Rainy

2024-01-05,30,Sunny

Open

weather.csv
~/

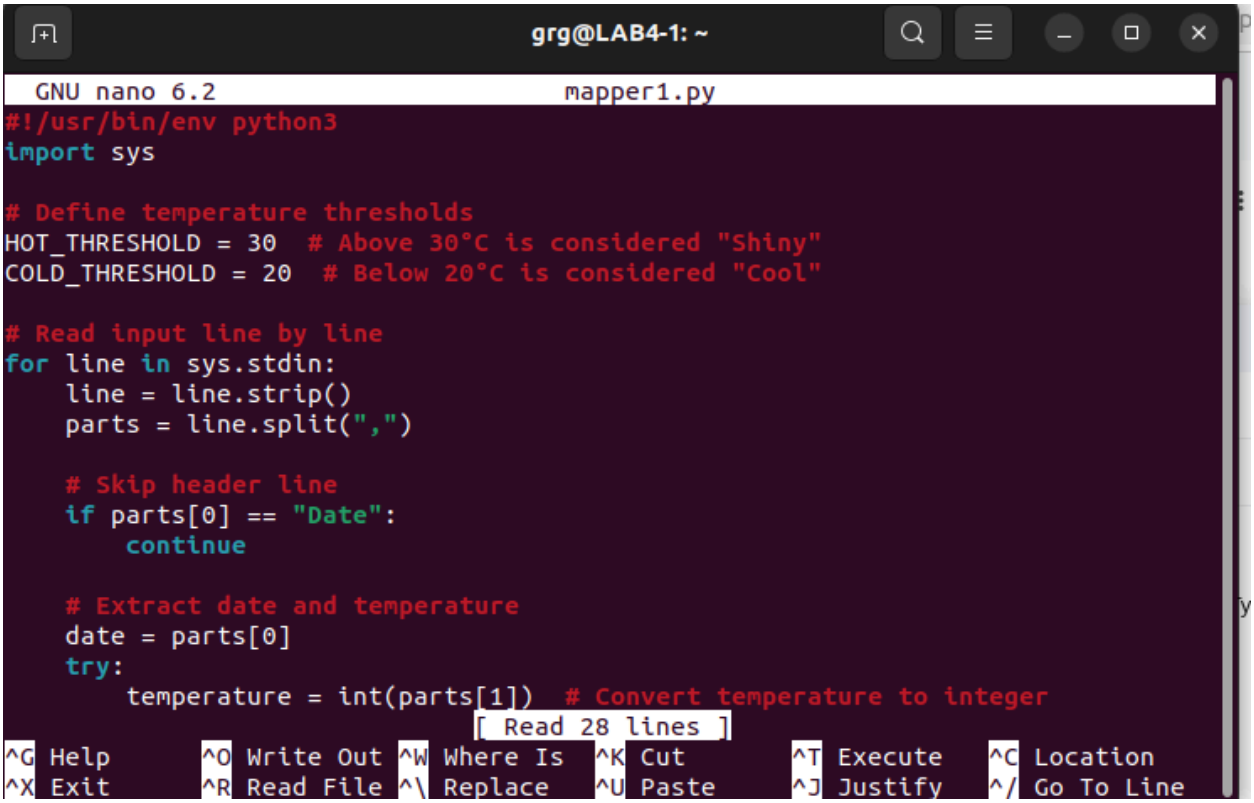
```
1 Date,Temperature,WeatherType
2 2024-01-01,35,Sunny
3 2024-01-02,18,Cloudy
4 2024-01-03,42,Sunny
5 2024-01-04,12,Rainy
6 2024-01-05,30,Sunny
```

INSERT INTO mapper1.py

```
#!/usr/bin/env python3
import sys

# Define temperature thresholds
HOT_THRESHOLD = 30  # Above 30°C is considered "Shiny"
COLD_THRESHOLD = 20  # Below 20°C is considered "Cool"

# Read input line by line
for line in sys.stdin:
    line = line.strip()
    parts = line.split(",")

    # Skip header line
    if parts[0] == "Date":
        continue

    # Extract date and temperature
    date = parts[0]
    try:
        temperature = int(parts[1])  # Convert temperature to integer

        # Categorize the day as "Shiny" or "Cool"
        if temperature > HOT_THRESHOLD:
            print(f"{date}\tShiny")
        elif temperature < COLD_THRESHOLD:
            print(f"{date}\tCool")
```

except ValueError:

continue # Skip lines with invalid data



After type the code press F2

INSERT INTO reducer.py

#!/usr/bin/env python3

import sys

Read input from standard input (sorted by Hadoop)

for line in sys.stdin:

line = line.strip()

try:

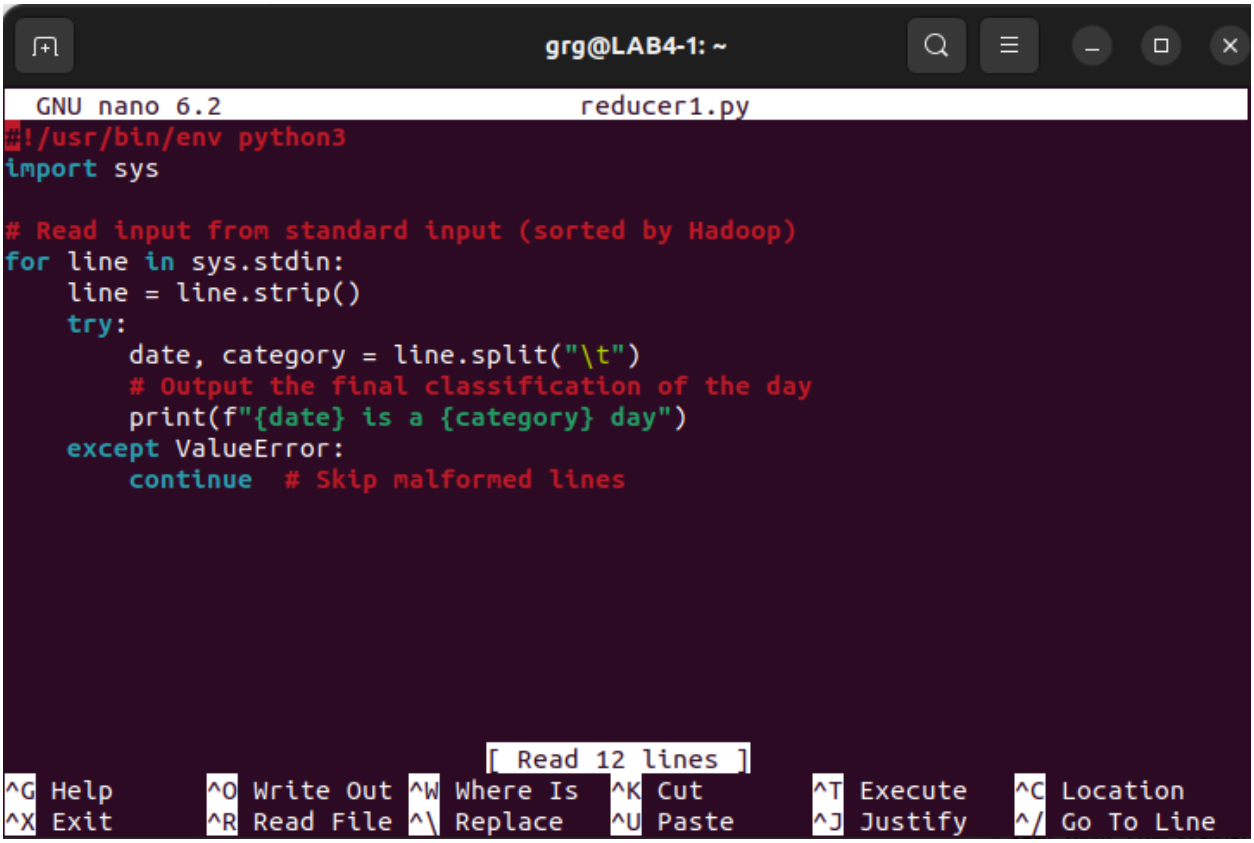
date, category = line.split("\t")

Output the final classification of the day

print(f"{date} is a {category} day")

except ValueError:

continue # Skip malformed lines



After type the code press F2

OUTPUT

```
grg@LAB4-1:~$ cat weather.csv | python3 mapper1.py | python3 reducer1.py > output.txt
grg@LAB4-1:~$ cat output.txt
2024-01-01 is a Shiny day
2024-01-02 is a Cool day
2024-01-03 is a Shiny day
2024-01-04 is a Cool day
grg@LAB4-1:~$
```

PROGRAM 17 Write a simple YARN client Application. Develop a Map Reduce program to calculate the frequency of a given word in given file.

TERMINAL

touch frequency.txt

nano mapper3.py

nano reducer3.py

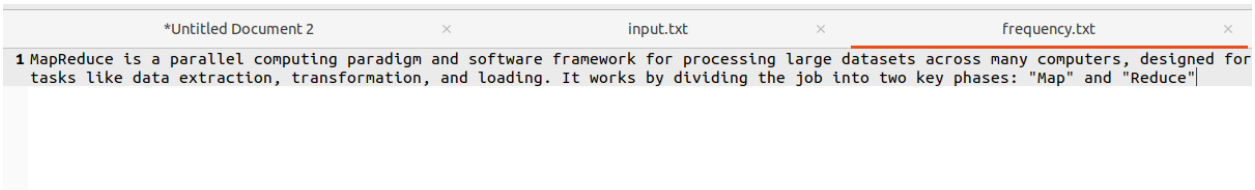
cat frequency.txt

cat frequency.txt | python3 mapper3.py | python3 reducer3.py > result.txt

cat result.txt

CREATE INTO frequency.txt —--TEXT EDITOR

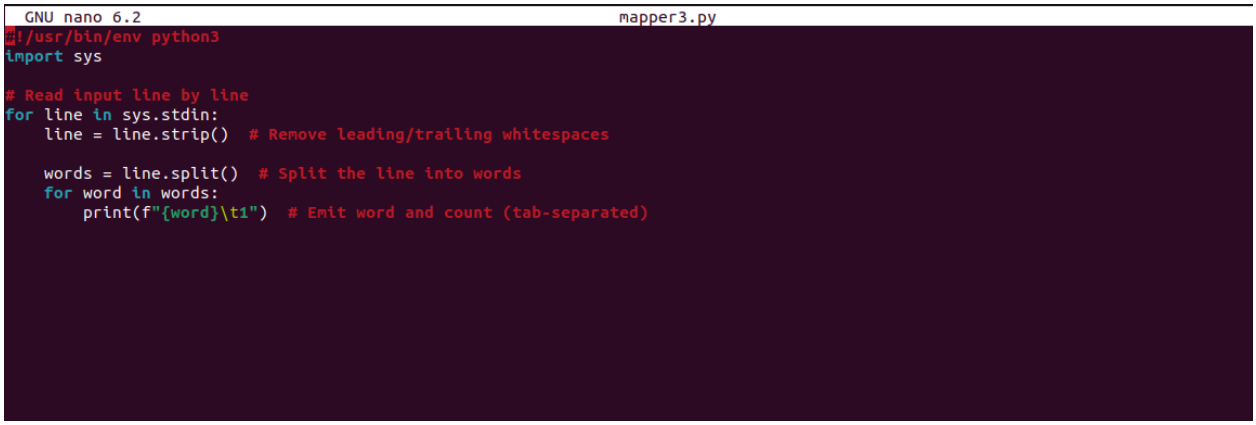
MapReduce is a parallel computing paradigm and software framework for processing large datasets across many computers, designed for tasks like data extraction, transformation, and loading. It works by dividing the job into two key phases: "Map" and "Reduce"



INSERT INTO mapper3.py

```
#!/usr/bin/env python3
import sys

# Read input line by line
for line in sys.stdin:
    line = line.strip()  # Remove leading/trailing whitespaces

    words = line.split()  # Split the line into words
    for word in words:
```

print(f"{word}\t1") # Emit word and count (tab-separated)



INSERT INTO reducer.py

```
#!/usr/bin/env python3
import sys

word_counts = {}  # Dictionary to store word counts

# Read input from mapper
for line in sys.stdin:
    word, count = line.strip().split("\t")  # Split word and count
    count = int(count)  # Convert count to integer

    # Aggregate counts
    if word in word_counts:
        word_counts[word] += count
    else:
        word_counts[word] = count

# Print final word frequencies
for word, count in word_counts.items():
    print(f"{word}\t{count}")
```

```
 GNU nano 6.2                                          reducer3.py
#!/usr/bin/env python3
import sys

word_counts = {}  # Dictionary to store word counts

# Read input from mapper
for line in sys.stdin:
    word, count = line.strip().split("\t")  # Split word and count
    count = int(count)  # Convert count to integer

    # Aggregate counts
    if word in word_counts:
        word_counts[word] += count
    else:
        word_counts[word] = count

# Print final word frequencies
for word, count in word_counts.items():
    print(f"{word}\t{count}")
```

OUTPUT

```
grg@LAB4-1:~$ touch frequency.txt
grg@LAB4-1:~$ nano mapper3.py
grg@LAB4-1:~$ nano reducer3.py
grg@LAB4-1:~$ cat frequency.txt
grg@LAB4-1:~$ cat frequency.txt
MapReduce is a parallel computing paradigm and software framework for processing large datasets across many computers, designed for
tasks like data extraction, transformation, and loading. It works by dividing the job into two key phases: "Map" and "Reduce"
grg@LAB4-1:~$ cat frequency.txt | python3 mapper3.py | python3 reducer3.py > result.txt
```

```
grg@LAB4-1:~$ cat result.txt
MapReduce       1
is      1
a       1
parallel        1
computing       1
paradigm        1
and     3
software        1
framework       1
for     2
processing      1
large   1
datasets        1
across  1
many    1
computers,      1
designed        1
tasks   1
like    1
data    1
extraction,     1
transformation, 1
loading.        1
It      1
works   1
by      1
dividing        1
the     1
job     1
into    1
two     1
key     1
phases: 1
"Map"   1
"Reduce"        1
```