TN Plantation Case Study

Name: Kartik Joshi

MID: M1074144

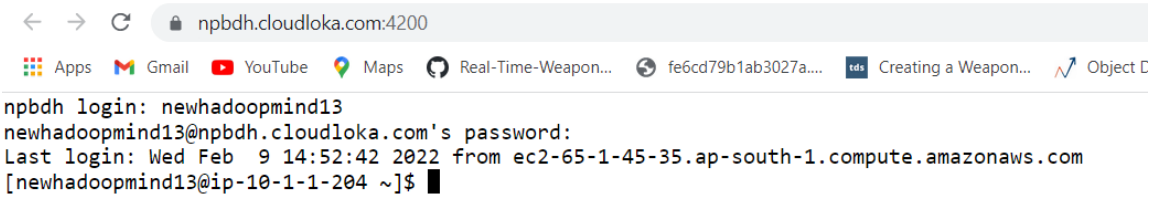
PROBLEM STATEMENT

- **Analyse which of the district have maximum and least production of**

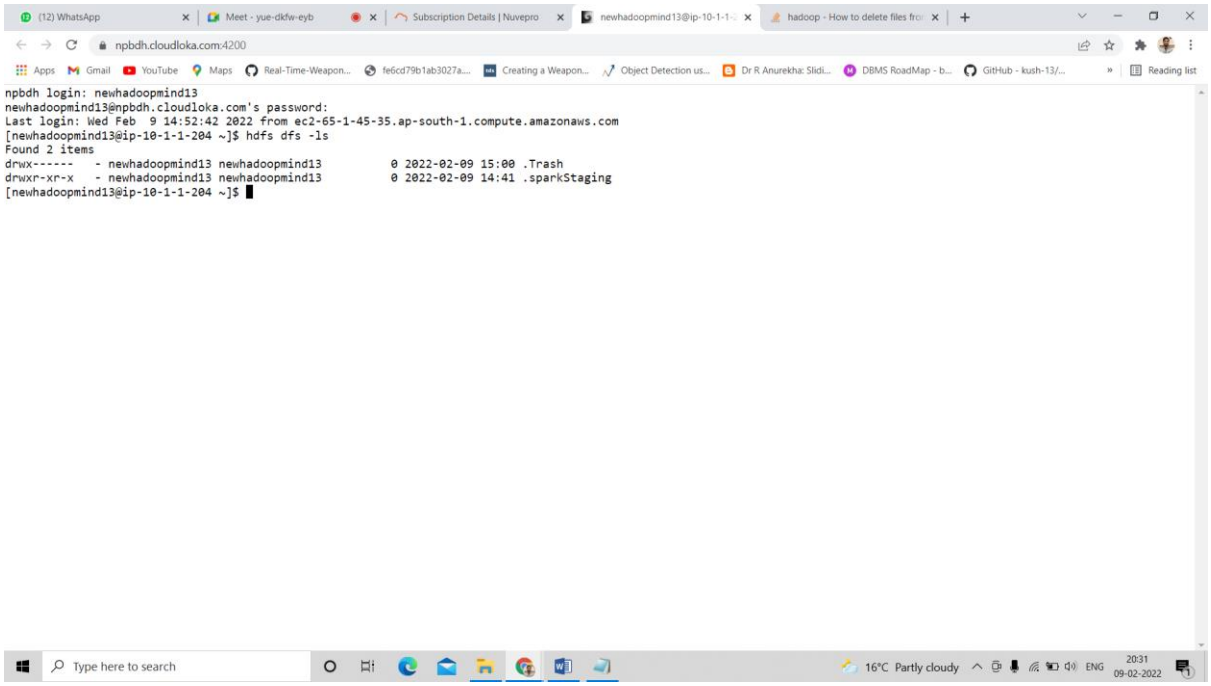
- **Tea**
- **Bamboo**
- **Rubber**

- **Total area of each plantation under all district**

- Open Cloud Web Shell(Nuvepro)



- Displaying the contents in hdfs – hdfs dfs -ls



- Putting the Datasets into Hadoop

hdfs dfs –put bamboo.txt

hdfs dfs –put tea.txt

hdfs dfs –put rubber.txt

```
[newhadoopmind13@ip-10-1-1-204 ~]$ hdfs dfs -put bamboo.txt
[newhadoopmind13@ip-10-1-1-204 ~]$ hdfs dfs -put tea.txt
[newhadoopmind13@ip-10-1-1-204 ~]$ hdfs dfs -put rubber.txt
[newhadoopmind13@ip-10-1-1-204 ~]$ hdfs dfs -ls
Found 5 items
drwx------   - newhadoopmind13 newhadoopmind13          0 2022-02-09 15:00 .Trash
drwxr-xr-x   - newhadoopmind13 newhadoopmind13          0 2022-02-09 14:41 .sparkStaging
-rw-r--r--   3 newhadoopmind13 newhadoopmind13        664 2022-02-09 15:02 bamboo.txt
-rw-r--r--   3 newhadoopmind13 newhadoopmind13        666 2022-02-09 15:02 rubber.txt
-rw-r--r--   3 newhadoopmind13 newhadoopmind13        697 2022-02-09 15:02 tea.txt
[newhadoopmind13@ip-10-1-1-204 ~]$
```

- Lets open a spark shell to implement our logic

Command – Spark-shell

```
[newhadoopmind13@ip-10-1-1-204 ~]$ spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context available as 'sc' (master = yarn, app id = application_1644201200461_0642).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.0-cdh6.2.1
      /_/

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_181)
Type in expressions to have them evaluated.
Type :help for more information.
```

- Importing required packages for use

Commands-

import org.apache.spark.sql.types._
import org.apache.spark.sql.Row;
import org.apache.spark.sql.functions.regexp_replace
import org.apache.spark.sql.DataFrameStatFunctions
import org.apache.spark.sql.functions._

```
scala> import spark.sql
import spark.sql

scala>

scala> import spark.implicits._
import spark.implicits._

scala>

scala> import org.apache.spark.sql.Row
import org.apache.spark.sql.Row

scala>

scala> import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.SparkSession

scala>

scala> import org.apache.spark._
import org.apache.spark._

scala>

scala> import org.apache.spark.rdd.RDD
import org.apache.spark.rdd.RDD
```

```
scala> import java.io.File
import java.io.File

scala>

scala> import scala.collection.mutable.ListBuffer
import scala.collection.mutable.ListBuffer

scala>

scala> import org.apache.spark.util.IntParam
import org.apache.spark.util.IntParam

scala>

scala> import org.apache.spark.sql.types._
import org.apache.spark.sql.types._

scala>

scala> import org.apache.spark.sql.Row;
import org.apache.spark.sql.Row

scala>

scala> import org.apache.spark.sql.functions.regexp_replace
import org.apache.spark.sql.functions.regexp_replace

scala>

scala> import org.apache.spark.sql.DataFrameStatFunctions
import org.apache.spark.sql.DataFrameStatFunctions

scala>

scala> import org.apache.spark.sql.functions._
import org.apache.spark.sql.functions._
```

- Creating a sql context variable

Command- *val sqlContext = new org.apache.spark.sql.SQLContext(sc)*

```
scala> val sqlContext = new org.apache.spark.sql.SQLContext(sc)
warning: there was one deprecation warning; re-run with -deprecation for details
sqlContext: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@2b4b3918
```

IMPORTING BAMBOO Dataset

- Now lets first create a schema to give our dataframe its structure:

Command- *val bSchema = StructType(List(StructField("Serial_Number", IntegerType, false), StructField("District", StringType, true), StructField("Area", StringType, true), StructField("Production", IntegerType, true)))*

```
scala> val bSchema = StructType(List(StructField("Serial_Number", IntegerType, false), StructField("District", StringType, true), StructField("Area", StringType, true),
 StructField("Production", StringType, true)))
bSchema: org.apache.spark.sql.types.StructType = StructType(StructField(Serial_Number,IntegerType,false), StructField(District,StringType,true), StructField(Area,String
Type,true), StructField(Production,StringType,true))
```

- Now creating the data frame for bamboo table

Command- val bambooData = spark.read.option("delimiter","|").schema(bSchema).csv(“bamboo.txt")

```
scala> val bambooData = spark.read.option("delimiter","|").schema(bSchema).csv("bamboo.txt")
bambooData: org.apache.spark.sql.DataFrame = [Serial_Number: int, District: string ... 2 more fields]
```

- Now cleaning out the district and area field using regex

Command - *val bambooDF = bambooData.withColumn("District", regexp_replace(col("District"), "[\\~, \\!, \\@, \\#, \\$, \\%, \\^, \\&, \\?, \\}, \\=]", ""))*

Command - *val bambooData = bambooDF.withColumn("Area", regexp_replace(col("Area"), "[^0-9]", ""))*

```
scala> val bambooDF = bambooData.withColumn("District", regexp_replace(col("District"), "[\\~, \\!, \\@, \\#, \\$, \\%, \\^, \\&, \\?, \\}, \\=]", ""))
bambooDF: org.apache.spark.sql.DataFrame = [Serial_Number: int, District: string ... 2 more fields]

scala>

scala> val bambooData = bambooDF.withColumn("Area", regexp_replace(col("Area"), "[^0-9]", ""))
bambooData: org.apache.spark.sql.DataFrame = [Serial_Number: int, District: string ... 2 more fields]
```

Now we can change the type of area field to integer

Command - *val bambooDF = bambooData.withColumn("Area", col("Area").cast(IntegerType))*

```
scala> val bambooDF = bambooData.withColumn("Area",col("Area").cast(IntegerType))
bambooDF: org.apache.spark.sql.DataFrame = [Serial_Number: int, District: string ... 2 more fields]
```

- Adding “PTY” column

Command - *val bambooData = bambooDF.withColumn("PTY", col("Production")/col("Area"))*

```
scala> val bambooData = bambooDF.withColumn("PTY", col("Production")/col("Area"))
bambooData: org.apache.spark.sql.DataFrame = [Serial_Number: int, District: string ... 3 more fields]

scala>
```

- Now lets round of the values of “PTY” column to 3 desimal places

Command- *val bambooDF = bambooData.withColumn("PTY", round(col("PTY") * 100 / 5) * 5 / 100)*

```
scala> val bambooDF = bambooData.withColumn("PTY", round(col("PTY") * 100 / 5) * 5 / 100)
bambooDF: org.apache.spark.sql.DataFrame = [Serial_Number: int, District: string ... 3 more fields]
```

Plantation type column for the easy identification of the type when we union

Command- *val bambooData = bambooDF.select(col("Serial_Number"), col("District"), col("Area"), col("Production"), col("PTY"), lit("Bamboo").as("Plantation_Type"))*

```
scala> val bambooData = bambooDF.select(col("Serial_Number"), col("District"), col("Area"), col("Production"), col("P
TY"), lit("Bamboo").as("Plantation_Type"))
bambooData: org.apache.spark.sql.DataFrame = [Serial_Number: int, District: string ... 4 more fields]
```

```
scala> bambooData.show()
+-------------+--------------+----+----------+-----+--------------+
|Serial_Number|      District|Area|Production|  PTY|Plantation_Type|
+-------------+--------------+----+----------+-----+--------------+
|            1|       Ariyalur| 112|      2258|20.15|        Bamboo|
|            2|    Coimbatore|   9|       181| 20.1|        Bamboo|
|            3|     Cuddalore| 142|      2863|20.15|        Bamboo|
|            4|    Dharmapuri|  11|       222| 20.2|        Bamboo|
|            5|      Dindigul|  19|       383|20.15|        Bamboo|
|            6|         Erode| 121|      2439|20.15|        Bamboo|
|            7|  Kancheepuram|  67|      1351|20.15|        Bamboo|
|            8|   Kanyakumari|   2|        40| 20.0|        Bamboo|
|            9|         Karur|   4|        81|20.25|        Bamboo|
|           10|   Krishnagiri|   4|        81|20.25|        Bamboo|
|           11|       Madurai|   4|        81|20.25|        Bamboo|
|           12|  Nagapattinam| 340|      6854|20.15|        Bamboo|
|           13|      Namakkal|   9|       181| 20.1|        Bamboo|
|           14|    Perambalur|  21|       423|20.15|        Bamboo|
|           15|   Pudukkottai|   3|        60| 20.0|        Bamboo|
|           16|Ramanadhapuram|null|      null| null|        Bamboo|
|           17|         Salem|  52|      1048|20.15|        Bamboo|
|           18|    Sivagangai|   3|        60| 20.0|        Bamboo|
|           19|     Thanjavur| 102|      2056|20.15|        Bamboo|
|           20|         Theni|  11|       222| 20.2|        Bamboo|
+-------------+--------------+----+----------+-----+--------------+
only showing top 20 rows
```

IMPORTING TEA dataset

- First we import the dataset

Command –

val textTea = sc.textFile("tea.txt")

```
scala> val textTea = sc.textFile("tea.txt")
textTea: org.apache.spark.rdd.RDD[String] = tea.txt MapPartitionsRDD[5] at textFile at <console>:62
```

- Now assign rdd to a string

Command - *val txt = textTea.first()*

```
scala> val txt = textTea.first()
txt: String = 1|Ariyalur|NA|NA|NA|2|Coimbatore|4|6|1.12|3|Cuddalore|NA|NA|NA|4|Dharmapuri|NA|NA|NA|5|Dindigul|NA|NA|N
A|6|Erode|NA|NA|NA|7|Kancheepuram|NA|NA|NA|8|Kanyakumari|26810|30027|1.12|9|Karur|NA|NA|NA|10|Krishnagiri|NA|NA|NA|11
|Madurai|NA|NA|NA|12|Nagapattinam|NA|NA|NA|13|Namakkal|NA|NA|NA|14|Perambalur|NA|NA|NA|15|Pudukkottai|NA|NA|NA|16|Ram
anadhapuram|NA|NA|NA|17|Salem|NA|NA|NA|18|Sivagangai|NA|NA|NA|19|Thanjavur|NA|NA|NA|20|Theni|NA|NA|NA|21|The Nilgiris
|159|200|1.12|22|Tiruvallur|NA|NA|NA|23|Thiruvarur|NA|NA|NA|24|Tiruvannamalai|NA|NA|NA|25|Thoothukudi|NA|NA|NA|26|Tir
upur|NA|NA|NA|27|Trichy|NA|NA|NA|28|Tirunelveli|43|48|1.12|29|Vellore|NA|NA|NA|30|Villupuram|NA|NA|NA|31|Virudhunagar
|NA|NA|NA

scala>
```

- *Giving line separation for every fifth delimiter*

Command –

val txtString = txt.replaceAll("(.?\\|){5}", "$0\n")*

```
scala> val txtString = txt.replaceAll("(.*?\\|){5}", "$0\n")
txtString: String =
1|Ariyalur|NA|NA|NA|
2|Coimbatore|4|6|1.12|
3|Cuddalore|NA|NA|NA|
4|Dharmapuri|NA|NA|NA|
5|Dindigul|NA|NA|NA|
6|Erode|NA|NA|NA|
7|Kancheepuram|NA|NA|NA|
8|Kanyakumari|26810|30027|1.12|
9|Karur|NA|NA|NA|
10|Krishnagiri|NA|NA|NA|
11|Madurai|NA|NA|NA|
12|Nagapattinam|NA|NA|NA|
13|Namakkal|NA|NA|NA|
14|Perambalur|NA|NA|NA|
15|Pudukkottai|NA|NA|NA|
16|Ramanadhapuram|NA|NA|NA|
17|Salem|NA|NA|NA|
18|Sivagangai|NA|NA|NA|
19|Thanjavur|NA|NA|NA|
20|Theni|NA|NA|NA|
21|The Nilgiris|159|200|1.12|
22|Tiruvallur|NA|NA|NA|
23|Thiruvarur|NA|NA|NA|
24|Tiruvannamalai|NA|NA|NA|
25|Thoothukudi|NA|NA|NA|
26|Tirupur|NA|NA|NA|
27|Trichy|NA|NA|NA|
28|Tirunelveli|43|48|1.12|
29|Vellore|NA|NA|NA|
30|Villupuram|NA|NA|NA|
31|Virudhunagar|NA|NA|NA

scala>
```

- *Now lets again convert the string into rdd*

Command *- val teaData = sc.parallelize(Seq(txtString))*

```
scala> val teaData = sc.parallelize(Seq(txtString))
teaData: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[4] at parallelize at <console>:63
```

- *Saving this rdd as a text file*

Command *- teaData.repartition(1).saveAsTextFile("newTea.txt")*

- *New schema of tea dataset*

Command *- val tSchema = new StructType().add("Serial_Number", IntegerType, false).add("District", StringType, true).add("Area", IntegerType, true).add("Production", IntegerType, true).add("PTY", FloatType, true)*

```
scala> teaData.repartition(1).saveAsTextFile("newTea.txt")

scala> val tSchema = new  StructType().add("Serial_Number", IntegerType, false).add("District", StringType, true).add("Area", IntegerType, true).add("Production", Integ
erType, true).add("PTY", FloatType, true)
tSchema: org.apache.spark.sql.types.StructType = StructType(StructField(Serial_Number,IntegerType,false), StructField(District,StringType,true), StructField(Area,Intege
rType,true), StructField(Production,IntegerType,true), StructField(PTY,FloatType,true))
```

- *Now we import data from newText file*

Command *- val teaData = spark.read.option("delimiter", "|").schema(tSchema).csv("newTea.txt/part-00000")*

```
scala> val teaData = spark.read.option("delimiter", "|").schema(tSchema).csv("newTea.txt/part-00000")
teaData: org.apache.spark.sql.DataFrame = [Serial_Number: int, District: string ... 3 more fields]
```

- Adding “Plantation_Type” column

Command *- val teaDF = teaData.select(col("Serial_Number"), col("District"), col("Area"), col("Production"), col("PTY"), lit("Tea").as("Plantation_Type"))*

```
scala> val teaDF = teaData.select(col("Serial_Number"), col("District"), col("Area"), col("Production"), col("PTY"),
lit("Tea").as("Plantation_Type"))
teaDF: org.apache.spark.sql.DataFrame = [Serial_Number: int, District: string ... 4 more fields]
```

```
scala> teaDF.show()
+-------------+--------------+-----+----------+----+--------------+
|Serial_Number|      District| Area|Production| PTY|Plantation_Type|
+-------------+--------------+-----+----------+----+--------------+
|            1|      Ariyalur| null|      null|null|           Tea|
|            2|    Coimbatore|    4|         6|1.12|           Tea|
|            3|     Cuddalore| null|      null|null|           Tea|
|            4|    Dharmapuri| null|      null|null|           Tea|
|            5|      Dindigul| null|      null|null|           Tea|
|            6|         Erode| null|      null|null|           Tea|
|            7|  Kancheepuram| null|      null|null|           Tea|
|            8|   Kanyakumari|26810|     30027|1.12|           Tea|
|            9|         Karur| null|      null|null|           Tea|
|           10|   Krishnagiri| null|      null|null|           Tea|
|           11|       Madurai| null|      null|null|           Tea|
|           12|  Nagapattinam| null|      null|null|           Tea|
|           13|      Namakkal| null|      null|null|           Tea|
|           14|    Perambalur| null|      null|null|           Tea|
|           15|   Pudukkottai| null|      null|null|           Tea|
|           16|Ramanadhapuram| null|      null|null|           Tea|
|           17|         Salem| null|      null|null|           Tea|
|           18|    Sivagangai| null|      null|null|           Tea|
|           19|     Thanjavur| null|      null|null|           Tea|
|           20|         Theni| null|      null|null|           Tea|
+-------------+--------------+-----+----------+----+--------------+
only showing top 20 rows
```

Importing the rubber schema

- Making a data frame from rubber

Command- *val rubberData = spark.read.option("delimiter","|").option("inferSchema", "true").option("header","true").csv("rubber.txt")*

```
scala> val rubberData = spark.read.option("delimiter","|").option("inferSchema", "true").option("header","true").csv("rubber.txt")
rubberData: org.apache.spark.sql.DataFrame = [Serial_Number: int, District: string ... 2 more fields]
```

- Adding column “PTY”

Command - *val rubber = rubberData.withColumn("PTY", col("Production")/col("Area"))*

Rounging off that column to 3 decimal places
*Command - val rubberDF = rubber.withColumn("PTY", round(col("PTY") * 100 / 5) * 5 / 100)*

Finally adding plantation type column for future use
Command - val rubberData = rubberDF.select(col("Serial_Number"), col("District"), col("Area"), col("Production"), col("PTY"), lit("Rubber").as("Plantation_Type"))

```
scala> val rubberDF = rubberData.withColumn("Area", col("Area").cast(IntegerType))
rubberDF: org.apache.spark.sql.DataFrame = [Serial_Number: int, District: string ... 2 more fields]

scala>

scala> val rubberData = rubberDF.withColumn("Production", col("Production").cast(IntegerType))
rubberData: org.apache.spark.sql.DataFrame = [Serial_Number: int, District: string ... 2 more fields]

scala>

scala> val rubber = rubberData.withColumn("PTY", col("Production")/col("Area"))
rubber: org.apache.spark.sql.DataFrame = [Serial_Number: int, District: string ... 3 more fields]

scala>

scala> val rubberDF = rubber.withColumn("PTY", round(col("PTY") * 100 / 5) * 5 / 100)
rubberDF: org.apache.spark.sql.DataFrame = [Serial_Number: int, District: string ... 3 more fields]

scala>

scala> val rubberData = rubberDF.select(col("Serial_Number"), col("District"), col("Area"), col("Production"), col("PTY"), lit("Rubber").as("Plantation_Type"))
rubberData: org.apache.spark.sql.DataFrame = [Serial_Number: int, District: string ... 4 more fields]
```

```
scala> rubberData.show(100)
+-------------+--------------+-----+----------+----+---------------+
|Serial_Number|      District| Area|Production| PTY|Plantation_Type|
+-------------+--------------+-----+----------+----+---------------+
|            1|      Ariyalur| null|      null|null|         Rubber|
|            2|    Coimbatore|    4|         6| 1.5|         Rubber|
|            3|     Cuddalore| null|      null|null|         Rubber|
|            4|    Dharmapuri| null|      null|null|         Rubber|
|            5|      Dindigul| null|      null|null|         Rubber|
|            6|         Erode| null|      null|null|         Rubber|
|            7|  Kancheepuram| null|      null|null|         Rubber|
|            8|   Kanyakumari|26810|     30027| 1.1|         Rubber|
|            9|         Karur| null|      null|null|         Rubber|
|           10|   Krishnagiri| null|      null|null|         Rubber|
|           11|       Madurai| null|      null|null|         Rubber|
|           12|  Nagapattinam| null|      null|null|         Rubber|
|           13|      Namakkal| null|      null|null|         Rubber|
|           14|    Perambalur| null|      null|null|         Rubber|
|           15|   Pudukkottai| null|      null|null|         Rubber|
|           16|Ramanadhapuram| null|      null|null|         Rubber|
|           17|         Salem| null|      null|null|         Rubber|
|           18|    Sivagangai| null|      null|null|         Rubber|
|           19|     Thanjavur| null|      null|null|         Rubber|
|           20|         Theni| null|      null|null|         Rubber|
|           21|  The Nilgiris|  159|       200|1.25|         Rubber|
|           22|    Tiruvallur| null|      null|null|         Rubber|
|           23|    Thiruvarur| null|      null|null|         Rubber|
|           24|Tiruvannamalai| null|      null|null|         Rubber|
|           25|   Thoothukudi| null|      null|null|         Rubber|
|           26|       Tirupur| null|      null|null|         Rubber|
|           27|        Trichy| null|      null|null|         Rubber|
|           28|   Tirunelveli|   43|        48| 1.1|         Rubber|
|           29|       Vellore| null|      null|null|         Rubber|
|           30|    Villupuram| null|      null|null|         Rubber|
|           31|  Virudhunagar| null|      null|null|         Rubber|
+-------------+--------------+-----+----------+----+---------------+
```

SOLUTIONS TO BUSINESS PROBLEM

Lets find the union to all three databases

Command - *val totalDF = bambooData.union(teaDF).union(rubberData).filter($"PTY".isNotNull)*

Null values have been filtered out we will cast the production and area column to integer type due to schema mismatch

Command - val totalData = totalDF.withColumn("Production", col("Production").cast(IntegerType))

Command - val totalDF = totalData.withColumn("Area", col("Area").cast(IntegerType))

Command – totalDF.createTempView(“Data”)

```
scala> val totalDF = bambooData.union(teaDF).union(rubberData).filter($"PTY".isNotNull)
totalDF: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Serial_Number: int, District: string ... 4 more fi
elds]

scala> val totalData = totalDF.withColumn("Production", col("Production").cast(IntegerType))
totalData: org.apache.spark.sql.DataFrame = [Serial_Number: int, District: string ... 4 more fields]

scala> val totalDF = totalData.withColumn("Area", col("Area").cast(IntegerType))
totalDF: org.apache.spark.sql.DataFrame = [Serial_Number: int, District: string ... 4 more fields]
```

Lets see the final data that we have taken union of-

```
scala> totalDF.show(100)
+-------------+--------------+-----+----------+------------------+--------------+
|Serial_Number|      District| Area|Production|               PTY|Plantation_Type|
+-------------+--------------+-----+----------+------------------+--------------+
|            1|      Ariyalur|  112|      2258|             20.15|         Bamboo|
|            2|    Coimbatore|    9|       181|              20.1|         Bamboo|
|            3|     Cuddalore|  142|      2863|             20.15|         Bamboo|
|            4|    Dharmapuri|   11|       222|              20.2|         Bamboo|
|            5|      Dindigul|   19|       383|             20.15|         Bamboo|
|            6|         Erode|  121|      2439|             20.15|         Bamboo|
|            7|  Kancheepuram|   67|      1351|             20.15|         Bamboo|
|            8|   Kanyakumari|    2|        40|              20.0|         Bamboo|
|            9|         Karur|    4|        81|             20.25|         Bamboo|
|           10|   Krishnagiri|    4|        81|             20.25|         Bamboo|
|           11|       Madurai|    4|        81|             20.25|         Bamboo|
|           12|  Nagapattinam|  340|      6854|             20.15|         Bamboo|
|           13|      Namakkal|    9|       181|              20.1|         Bamboo|
|           14|    Perambalur|   21|       423|             20.15|         Bamboo|
|           15|   Pudukkottai|    3|        60|              20.0|         Bamboo|
|           17|         Salem|   52|      1048|             20.15|         Bamboo|
|           18|    Sivagangai|    3|        60|              20.0|         Bamboo|
|           19|     Thanjavur|  102|      2056|             20.15|         Bamboo|
|           20|         Theni|   11|       222|              20.2|         Bamboo|
|           22|    Tiruvallur|   26|       524|             20.15|         Bamboo|
|           23|    Thiruvarur|   19|       383|             20.15|         Bamboo|
|           24|Tiruvannamalai|   75|      1512|             20.15|         Bamboo|
|           25|   Thoothukudi|   33|       665|             20.15|         Bamboo|
|           26|       Tirupur|    7|       141|             20.15|         Bamboo|
|           27|        Trichy|   52|      1048|             20.15|         Bamboo|
|           28|   Tirunelveli|   56|      1129|             20.15|         Bamboo|
|           29|       Vellore|   26|       524|             20.15|         Bamboo|
|           30|   Villup{uram|   30|       605|             20.15|         Bamboo|
|           31|  Virudhunagar|   10|       202|              20.2|         Bamboo|
|            2|    Coimbatore|    4|         6|1.1200000047683716|            Tea|
|            8|   Kanyakumari|26810|     30027|1.1200000047683716|            Tea|
|           21|  The Nilgiris|  159|       200|1.1200000047683716|            Tea|
|           28|   Tirunelveli|   43|        48|1.1200000047683716|            Tea|
|            2|    Coimbatore|    4|         6|               1.5|         Rubber|
|            8|   Kanyakumari|26810|     30027|               1.1|         Rubber|
|           21|  The Nilgiris|  159|       200|              1.25|         Rubber|
|           28|   Tirunelveli|   43|        48|               1.1|         Rubber|
+-------------+--------------+-----+----------+------------------+--------------+


scala>
```

Lets find the district having max production of bamboo

Command - *val maxBambooDF = sqlContext.sql("SELECT District, Production FROM Data WHERE Production IN (SELECT max(Production) FROM Data WHERE Plantation_Type='Bamboo')")*

Command – maxBambooDF.show()

```
scala> totalDF.createTempView("Data");

scala> val maxBambooDF = sqlContext.sql("SELECT District, Production FROM Data WHERE Production IN (SELECT max(Produc
tion) FROM Data WHERE Plantation_Type='Bamboo' )")
maxBambooDF: org.apache.spark.sql.DataFrame = [District: string, Production: int]

scala> maxBambooDF.show()
+------------+----------+
|    District|Production|
+------------+----------+
|Nagapattinam|      6854|
+------------+----------+
```

Lets calculate minimum bamboo production

Command - *val minBambooDF = sqlContext.sql("SELECT District, Production FROM Data WHERE Production IN (SELECT min(Production) FROM Data WHERE Plantation_Type='Bamboo')")*

Command – minBambooDF.show()

```
scala> val minBambooDF = sqlContext.sql("SELECT District, Production FROM Data WHERE Production IN (SELECT min(Produc
tion) FROM Data WHERE Plantation_Type='Bamboo' )")
minBambooDF: org.apache.spark.sql.DataFrame = [District: string, Production: int]

scala> minBambooDF.show()
+-----------+----------+
|   District|Production|
+-----------+----------+
|Kanyakumari|        40|
+-----------+----------+
```

Lets find district with maximum tea production:

Command - *val maxTeaDF = sqlContext.sql("SELECT District, Production FROM Data WHERE Production IN (SELECT max(Production) FROM Data WHERE Plantation_Type='Tea')")*

```
scala> val maxTeaDF = sqlContext.sql("SELECT District, Production FROM Data WHERE Production IN (SELECT max(Productio
n) FROM Data WHERE Plantation_Type='Tea' )")
maxTeaDF: org.apache.spark.sql.DataFrame = [District: string, Production: int]

scala> maxTeaDF.distinct().show()
+-----------+----------+
|   District|Production|
+-----------+----------+
|Kanyakumari|     30027|
+-----------+----------+
```

Lets find district with minimum tea production:

Command - *val minTeaDF = sqlContext.sql("SELECT District, Production FROM Data WHERE Production IN (SELECT min(Production) FROM Data WHERE Plantation_Type='Tea')")*

```
scala> val minTeaDF = sqlContext.sql("SELECT District, Production FROM Data WHERE Production IN (SELECT min(Productio
n) FROM Data WHERE Plantation_Type='Tea' )")
minTeaDF: org.apache.spark.sql.DataFrame = [District: string, Production: int]

scala> minTeaDF.distinct().show()
+----------+----------+
|  District|Production|
+----------+----------+
|Coimbatore|         6|
+----------+----------+
```

Lets find district with maximum rubber production:

Command - *val maxRubberDF = sqlContext.sql("SELECT District, Production FROM Data WHERE Production IN (SELECT max(Production) FROM Data WHERE Plantation_Type='Rubber')")*

```
scala> val maxRubberDF = sqlContext.sql("SELECT District, Production FROM Data WHERE Production IN (SELECT max(Produc
tion) FROM Data WHERE Plantation_Type='Rubber' )")
maxRubberDF: org.apache.spark.sql.DataFrame = [District: string, Production: int]

scala> maxRubberDF.distinct().show()
+-----------+----------+
|   District|Production|
+-----------+----------+
|Kanyakumari|     30027|
+-----------+----------+
```

Lets find district with minimum rubber production:

Command - *val minRubberDF = sqlContext.sql("SELECT District, Production FROM Data WHERE Production IN (SELECT min(Production) FROM Data WHERE Plantation_Type='Rubber')")*

```
scala> val minRubberDF = sqlContext.sql("SELECT District, Production FROM Data WHERE Production IN (SELECT min(Produc
tion) FROM Data WHERE Plantation_Type='Rubber' )")
minRubberDF: org.apache.spark.sql.DataFrame = [District: string, Production: int]

scala> minRubberDF.distinct().show()
+----------+----------+
|  District|Production|
+----------+----------+
|Coimbatore|         6|
+----------+----------+
```

Total area under each plantataion:

Bamboo
Command- *val BambooArea = sqlContext.sql("SELECT sum(Area) as Total_Area FROM Data WHERE Plantation_Type='Bamboo'")*

```
scala> val BambooArea = sqlContext.sql("SELECT sum(Area) as Total_Area FROM Data WHERE Plantation_Type='Bamboo'")
BambooArea: org.apache.spark.sql.DataFrame = [Total_Area: bigint]

scala> BambooArea.show()
+----------+
|Total_Area|
+----------+
|      1370|
+----------+
```

Tea
Command - val TeaArea = sqlContext.sql("SELECT sum(Area) as Total_Area FROM Data WHERE Plantation_Type='Tea'")

```
scala> val TeaArea = sqlContext.sql("SELECT sum(Area) as Total_Area FROM Data WHERE Plantation_Type='Tea'")
TeaArea: org.apache.spark.sql.DataFrame = [Total_Area: bigint]

scala> TeaArea.show()
+----------+
|Total_Area|
+----------+
|     27016|
+----------+
```

Rubber

Command - val RubberArea = sqlContext.sql("SELECT sum(Area) as Total_Area FROM Data WHERE Plantation_Type='Rubber'")

```
scala> val RubberArea = sqlContext.sql("SELECT sum(Area) as Total_Area FROM Data WHERE Plantation_Type='Rubber'")
RubberArea: org.apache.spark.sql.DataFrame = [Total_Area: bigint]

scala> RubberArea.show()
+----------+
|Total_Area|
+----------+
|     27016|
+----------+
```

PROBLEM STATEMENT

- **Identify whether there are any relations between bamboo, tea and rubber plantation of each of the district**

First we extract district and PTY coloumn from each of the plantation data frames , and rename the columns with PTY_PlantationName

Command - *val df = bambooDF.select("District", "PTY")*

Copies the column and deletes the PTY existing

Command - val newDF = df.withColumn("PTY_Bamboo", col("PTY")).drop("PTY")

```
scala> val df = bambooDF.select("District", "PTY")
df: org.apache.spark.sql.DataFrame = [District: string, PTY: double]

scala> val newDF = df.withColumn("PTY_Bamboo", col("PTY")).drop("PTY")
newDF: org.apache.spark.sql.DataFrame = [District: string, PTY_Bamboo: double]
```

Add the new DF to the old DF which had extracted data

Command - *val newDF = df.as("df").join(teaData.as("teaData"), df("District") === teaData("District")).select("df.District", "df.PTY_Bamboo", "teaData.PTY")*

```
scala> val newDF = df.as("df").join(teaData.as("teaData"), df("District") === teaData("District")).select("df.Distric
t", "df.PTY_Bamboo", "teaData.PTY")
newDF: org.apache.spark.sql.DataFrame = [District: string, PTY_Bamboo: double ... 1 more field]
```

Renaming the column for tea also
Command - *val df = newDF.withColumn("PTY_Tea", col("PTY")).drop("PTY")*

```
scala> df.show()
+--------------+----------+-------+
|      District|PTY_Bamboo|PTY_Tea|
+--------------+----------+-------+
|      Ariyalur|     20.15|   null|
|    Coimbatore|      20.1|   1.12|
|     Cuddalore|     20.15|   null|
|    Dharmapuri|      20.2|   null|
|      Dindigul|     20.15|   null|
|         Erode|     20.15|   null|
|  Kancheepuram|     20.15|   null|
|   Kanyakumari|      20.0|   1.12|
|         Karur|     20.25|   null|
|   Krishnagiri|     20.25|   null|
|       Madurai|     20.25|   null|
|  Nagapattinam|     20.15|   null|
|      Namakkal|      20.1|   null|
|    Perambalur|     20.15|   null|
|   Pudukkottai|      20.0|   null|
|Ramanadhapuram|      null|   null|
|         Salem|     20.15|   null|
|    Sivagangai|      20.0|   null|
|     Thanjavur|     20.15|   null|
|         Theni|      20.2|   null|
+--------------+----------+-------+
only showing top 20 rows
```

Adding now rubber
Command - val newDF = df.as("df").join(rubberDF.as("rubberDF"), df("District") === rubberDF("District")).select("df.District", "df.PTY_Bamboo", "df.PTY_Tea", "rubberDF.PTY")

Command - *val df = newDF.withColumn("PTY_Rubber", col("PTY")).drop("PTY")*

```
scala> val newDF = df.as("df").join(rubberDF.as("rubberDF"), df("District") === rubberDF("District")).select("df.Dist
rict", "df.PTY_Bamboo", "df.PTY_Tea", "rubberDF.PTY")
newDF: org.apache.spark.sql.DataFrame = [District: string, PTY_Bamboo: double ... 2 more fields]

scala> val df = newDF.withColumn("PTY_Rubber", col("PTY")).drop("PTY")
df: org.apache.spark.sql.DataFrame = [District: string, PTY_Bamboo: double ... 2 more fields]

scala> df.show()
+--------------+----------+-------+----------+
|      District|PTY_Bamboo|PTY_Tea|PTY_Rubber|
+--------------+----------+-------+----------+
|      Ariyalur|     20.15|   null|      null|
|    Coimbatore|      20.1|   1.12|       1.5|
|     Cuddalore|     20.15|   null|      null|
|    Dharmapuri|      20.2|   null|      null|
|      Dindigul|     20.15|   null|      null|
|         Erode|     20.15|   null|      null|
|  Kancheepuram|     20.15|   null|      null|
|   Kanyakumari|      20.0|   1.12|       1.1|
|         Karur|     20.25|   null|      null|
|   Krishnagiri|     20.25|   null|      null|
|       Madurai|     20.25|   null|      null|
|  Nagapattinam|     20.15|   null|      null|
|      Namakkal|      20.1|   null|      null|
|    Perambalur|     20.15|   null|      null|
|   Pudukkottai|      20.0|   null|      null|
|Ramanadhapuram|      null|   null|      null|
|         Salem|     20.15|   null|      null|
|    Sivagangai|      20.0|   null|      null|
|     Thanjavur|     20.15|   null|      null|
|         Theni|      20.2|   null|      null|
+--------------+----------+-------+----------+
only showing top 20 rows
```

For finding co-relation

Command – df.stat.corr("PTY_Rubber", "PTY_Bamboo")

Command - df.stat.corr("PTY_Rubber", "PTY_Tea")

Command - df.stat.corr("PTY_Bamboo", " PTY_Tea")

```
scala> df.stat.corr("PTY_Rubber", "PTY_Bamboo")
res23: Double = 0.05777378786811142

scala> df.stat.corr("PTY_Rubber", "PTY_Tea")
res24: Double = 0.987213582608028

scala> df.stat.corr("PTY_Bamboo", "PTY_Tea")
res25: Double = 0.058355603575429404
```

There is one particular set of variables that are particularly interesting. The correlation between PTY_Tea and PTY_Rubber is very close to 1.0.

A correlation of 1.0 means there is a perfect positive relationship between the two variables. For a positive increase in one variable, there is also a positive increase in the second variable.

PTY_Tea and PTY_Rubber being so close to 1.0 implies that when productivity in one of the plantations increases, the productivity of the other plantation increases as well.