CHILDNAME ANALYSIS - SCALA

* <https://www.ssa.gov/OACT/babynames/limits.html>. Make sure that you download the State-specific data.

Upon unzipping the file, you will have 51 text files organized by their two-letter state code (50 states plus the District of Columbia).

Here are each of the columns in each file:

* State (2 letter code)
* Sex (F for female, M for male)
* Year (4 digit number)
* Name
* Count (number of times a baby was born in this state with this name)

Unfortunately there is no header to these files. I provided my own column headers to each of the files required below: "state","sex","year","name","count"

## Directions

1. Open 4 different states. The arbitrary requirement here is that three of the states much touch a fourth state. This way we can study a geographic area. You may add your headers manually to each file.
2. Concatenate each of your four data sets into a single datasets.
3. How many females were born in your four selected states? How many males were born? Use a pivot table to create your answer and select the appropriate aggregation method.
4. How many people were born in each state? Again, use a pivot table.
5. What are the top 5 most popular names for any sex? Again, use a pivot table.
6. What are the top 5 most popular female names? (Use a pivot table to answer this.)
7. What are the top 5 most popular male names? (Use a pivot table to answer this.)

Question 1)

scala> val TN=spark.read.option("header","true").csv("/home/farjana/Data/TN.TXT")

TN: org.apache.spark.sql.DataFrame = [state: string, sex: string ... 3 more fields]

scala> TN.show(5)

+-----+---+----+---------+-----+

|state|sex|year| name|count|

+-----+---+----+---------+-----+

| TN| F|1910| Mary| 735|

| TN| F|1910| Ruby| 168|

| TN| F|1910| Annie| 163|

| TN| F|1910| Ruth| 163|

| TN| F|1910|Elizabeth| 154|

+-----+---+----+---------+-----+

only showing top 5 rows

scala> val MS=spark.read.option("header","true").csv("/home/farjana/Data/MS.TXT")

MS: org.apache.spark.sql.DataFrame = [state: string, sex: string ... 3 more fields]

scala> MS.show(5)

+-----+---+----+------+-----+

|state|sex|year| name|count|

+-----+---+----+------+-----+

| MS| F|1910| Mary| 762|

| MS| F|1910| Annie| 354|

| MS| F|1910|Willie| 208|

| MS| F|1910|Mattie| 178|

| MS| F|1910|Lillie| 158|

+-----+---+----+------+-----+

only showing top 5 rows

scala> val GA=spark.read.option("header","true").csv("/home/farjana/Data/GA.TXT")

GA: org.apache.spark.sql.DataFrame = [state: string, sex: string ... 3 more fields]

scala> GA.show(5)

+-----+---+----+------+-----+

|state|sex|year| name|count|

+-----+---+----+------+-----+

| GA| F|1910| Mary| 841|

| GA| F|1910| Annie| 553|

| GA| F|1910|Mattie| 320|

| GA| F|1910| Ruby| 279|

| GA| F|1910|Willie| 275|

+-----+---+----+------+-----+

only showing top 5 rows

scala> val AL=spark.read.option("header","true").csv("/home/farjana/Data/AL.TXT")

AL: org.apache.spark.sql.DataFrame = [state: string, sex: string ... 3 more fields]

scala> AL.show(5)

+-----+---+----+------+-----+

|state|sex|year| name|count|

+-----+---+----+------+-----+

| AL| F|1910| Mary| 875|

| AL| F|1910| Annie| 482|

| AL| F|1910|Willie| 257|

| AL| F|1910|Mattie| 232|

| AL| F|1910| Ruby| 204|

+-----+---+----+------+-----+

only showing top 5 rows

Question 2)

scala> val combine2=TN.union(MS)

combine2: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [state: string, sex: string ... 3 more fields]

scala> combine2.count()

res4: Long = 249794

scala> TN.count()

res5: Long = 137524

scala> MS.count()

res6: Long = 112270

scala> val combine3=combine2.union(GA)

combine3: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [state: string, sex: string ... 3 more fields]

scala> combine3.count()

res7: Long = 428220

scala> GA.count()

res8: Long = 178426

scala> val combinedall=combine3.union(AL)

combinedall: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [state: string, sex: string ... 3 more fields]

scala> combinedall.count()

res9: Long = 560285

scala> AL.count()

res10: Long = 132065

Question 3)

scala> val pivoted=combinedall.groupBy("sex").pivot("sex").agg("count"->"sum").show()

+---+-----------+-----------+

|sex| F| M|

+---+-----------+-----------+

| F|1.2168859E7| null|

| M| null|1.2892503E7|

+---+-----------+-----------+

pivoted: Unit = ()

Question 4)

scala> val pivoted=combinedall.groupBy("state").pivot("state").agg("count"->"sum").show()

+-----+---------+---------+---------+---------+

|state| AL| GA| MS| TN|

+-----+---------+---------+---------+---------+

| MS| null| null|4068851.0| null|

| AL|5815853.0| null| null| null|

| TN| null| null| null|6583232.0|

| GA| null|8593426.0| null| null|

+-----+---------+---------+---------+---------+

pivoted: Unit = ()

Question 5)

scala> val convert=combinedall.withColumn("count",$"count".cast(sql.types.IntegerType))

convert: org.apache.spark.sql.DataFrame = [state: string, sex: string ... 3 more fields]

scala> convert.groupBy("name").sum("count").withColumnRenamed("sum(count)","total").sort(desc("total")).limit(5).show()

+-------+------+

| name| total|

+-------+------+

| James|631285|

| Mary|435135|

|William|417741|

| John|386511|

| Robert|353398|

+-------+------+

Question 6)

scala> convert.where("sex='F'").groupBy("name","sex").sum("count").withColumnRenamed("sum(count)","total").sort(desc("total")).limit(5).show(5)

+---------+---+------+

| name|sex| total|

+---------+---+------+

| Mary| F|433426|

| Betty| F|131289|

| Linda| F|124409|

|Elizabeth| F|120112|

| Dorothy| F|114491|

+---------+---+------+

Question 7)

scala> convert.where("sex='M'").groupBy("name","sex").sum("count").withColumnRenamed("sum(count)","total").sort(desc("total")).limit(5).show(5)

+-------+---+------+

| name|sex| total|

+-------+---+------+

| James| M|627826|

|William| M|416459|

| John| M|384564|

| Robert| M|351699|

|Charles| M|259378|

+-------+---+------+