Case_Study_-_V Hospital_Analysis_in_US

DESCRIPTION –

Dataset Description

- **DRG Definition**: The code and description identifying the MS-DRG. MS-DRGs are a classification system that groups similar clinical conditions (diagnoses) and procedures furnished by the hospital during their stay.
- **Provider Id**: The CMS Certification Number (CCN) assigned to the Medicare-certified hospital facility.
- **Provider Name**: The name of the provider.
- **Provider Street Address**: The provider's street address.
- **Provider City**: The city where the provider is located.
- **Provider State**: The state where the provider is located.
- **Provider Zip Code**: The provider's zip code.
- **Provider HRR**: The Hospital Referral Region (HRR) where the provider is located.
- **Total Discharges**: The number of discharges billed by the provider for inpatient hospital services.
- **Average Covered Charges**: The provider's average charge for services covered by Medicare for all discharges in the MS-DRG. These will vary from hospital to hospital because of the differences in hospital charge structures.
- **Average Total Payments**: The average total payments to all providers for the MS-DRG including the MSDRG amount, teaching, disproportionate share, capital, and outlier payments for all cases. Also included in the average total payments are co-payment and deductible amounts that the patient is responsible for and any additional payments by third parties for coordination of benefits.
- **Average Medicare Payments**: The average amount that Medicare pays to the provider for Medicare's share of the MS-DRG. Average Medicare payment amounts include the MS-DRG amount, teaching, disproportionate share, capital, and outlier payments for all cases. Medicare payments DO NOT include beneficiary co-payments and deductible amounts nor any additional payments from third parties for coordination of benefits.

OBJECTIVES –

1. Load file into spark.
2. What is the average amount of **AverageCoveredCharges** per state.
3. Find out the **AverageTotalPayments** charges per state.
4. Find out the **AverageMedicarePayments** charges per state.
5. Find out the total number of **Discharges** per state and for each disease.
6. Sort the output in descending order of **totalDischarges**.

SOLUTION –

The dataset (inpatientCharges.csv) consists of 163065 records and we will evaluating few problem statements using Spark SQL.

- Setting up Spark context.

```
val spark =
org.apache.spark.sql.SparkSession.builder.master("local").appName("HospitalAnalysis
").getOrCreate;
```

- Load a CSV file directly into the Spark SQL context as follows.

```
val df = spark.read.format("com.databricks.spark.csv").option("header", "true")
   .option("inferSchema", "true").
   load("C:\\Users\\prave\\Desktop\\Spark\\inpatientCharges.csv")
```

We have loaded all the CSV data as a DataFrame into Spark SQL. Here, we have used inferschema as an option so it will automatically infer the data type of the columns.

- Schema of the data frame.

```
df.printSchema()
```

HospitalAnalysis › main(args: Array[String])

HospitalAnalysis

```
root
 |-- DRGDefinition: string (nullable = true)
 |-- ProviderId: integer (nullable = true)
 |-- ProviderName: string (nullable = true)
 |-- ProviderStreetAddress: string (nullable = true)
 |-- ProviderCity: string (nullable = true)
 |-- ProviderState: string (nullable = true)
 |-- ProviderZipCode: integer (nullable = true)
 |-- HospitalReferralRegionDescription: string (nullable = true)
 |-- TotalDischarges: integer (nullable = true)
 |-- AverageCoveredCharges: double (nullable = true)
 |-- AverageTotalPayments: double (nullable = true)
 |-- AverageMedicarePayments: double (nullable = true)
```

- Save data in a table by registering in a temp table “Hospital_Charges”

```
df.registerTempTable("Hospital_Charges")
```

Problem Statement 1: Find the amount of **Average Covered Charges** per state.

df.groupBy("ProviderState").avg("AverageCoveredCharges").show

```
df.groupBy( col1 = "ProviderState").avg( colNames = "AverageCoveredCharges").show
HospitalAnalysis › main(args: Array[String])
HospitalAnalysis ×
18/11/26 14:28:49 INFO DAGScheduler: Job 6 finished: show at HospitalAnalysis.scala
18/11/26 14:28:49 INFO CodeGenerator: Code generated in 52.025488 ms
+-------------+-------------------------+
|ProviderState|avg(AverageCoveredCharges)|
+-------------+-------------------------+
|           AZ|        41200.063019992995|
|           SC|         35862.49456269756|
|           LA|        33085.372791542846|
|           MN|         27894.36182060388|
|           NJ|         66125.68627434729|
|           DC|         40116.66365800864|
|           OR|        27390.111870669723|
|           VA|        29222.000487072903|
|           RI|        29942.701122448976|
|           KY|         24523.80716940223|
|           WY|         28700.59862348178|
|           NH|        27059.020801944105|
|           MI|        24124.247209817277|
|           NV|         61047.11541597337|
|           WI|        26149.325331686607|
|           ID|        25565.547041742288|
|           CA|           67508.616535517|
|           CT|          31318.4101143709|
|           NE|        31736.427824858758|
|           MT|        22670.015237154144|
+-------------+-------------------------+
only showing top 20 rows
```

Problem Statement 2: Find the amount of **Average Total Payments** charges per state**.**

df.groupBy("ProviderState").avg("AverageTotalPayments").show

```
df.groupBy( col1 = "ProviderState").avg( colNames = "AverageTotalPayments").show
```

HospitalAnalysis › main(args: Array[String])

HospitalAnalysis

```
+-------------+-------------------------+
|ProviderState|avg(AverageTotalPayments)|
+-------------+-------------------------+
|           AZ|       10154.528211153991|
|           SC|        9132.420758693366|
|           LA|         8638.66257680871|
|           MN|        9948.236962699833|
|           NJ|        10678.98864691253|
|           DC|       12998.029415584406|
|           OR|       10436.192863741335|
|           VA|         8887.75217682364|
|           RI|       10509.566853741484|
|           KY|         8278.58884484363|
|           WY|       11398.485910931167|
|           NH|        9289.661822600248|
|           MI|        9754.420405978948|
|           NV|       10291.718028286188|
|           WI|        9270.705617501746|
|           ID|        9827.180090744107|
|           CA|       12629.668472137122|
|           CT|       11365.450671307795|
|           NE|        9331.682523540492|
|           MT|        9252.802766798422|
+-------------+-------------------------+
only showing top 20 rows
```

Problem Statement 3: Find the amount of **Average Medicare Payments** charges per state**.**

df.groupBy("ProviderState").avg("AverageMedicarePayments").show

```
df.groupBy( col1 = "ProviderState").avg( colNames = "AverageMedicarePayments").show
```

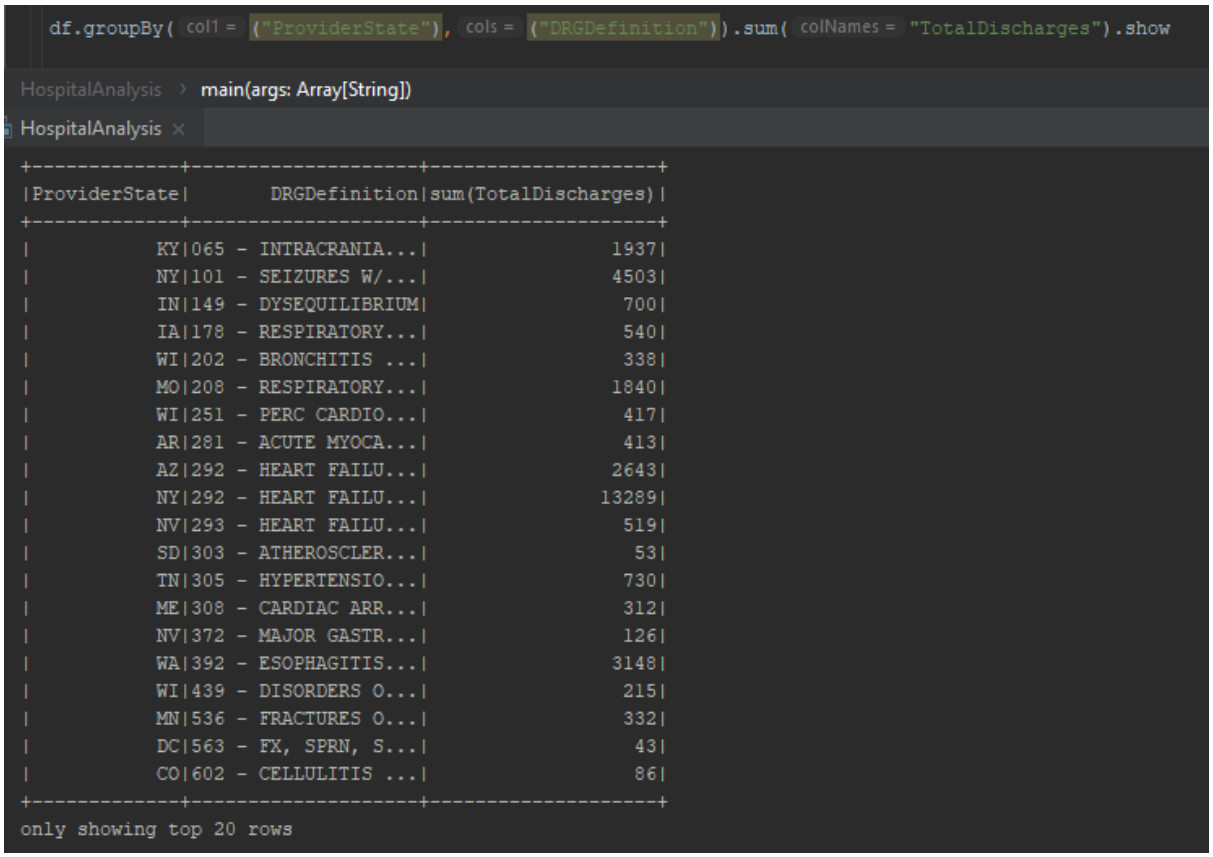
HospitalAnalysis main(args: Array[String])

HospitalAnalysis

```
+-------------+---------------------------+
|ProviderState|avg(AverageMedicarePayments)|
+-------------+---------------------------+
|           AZ|           8825.717239565045|
|           SC|            7876.33152441167|
|           LA|           7387.704625041281|
|           MN|           8619.214982238007|
|           NJ|           9586.940055946912|
|           DC|          11811.967705627709|
|           OR|           9035.259961508847|
|           VA|           7538.847006001846|
|           RI|           9317.939115646255|
|           KY|           7185.227810467647|
|           WY|           9539.392024291496|
|           NH|           8124.506852976913|
|           MI|           8662.157756043543|
|           NV|           8747.602828618963|
|           WI|           8002.597911079731|
|           ID|           8461.977513611617|
|           CA|          11494.381677893474|
|           CT|          10104.592943809059|
|           NE|          7992.6272504707995|
|           MT|           7981.088063241104|
+-------------+---------------------------+
only showing top 20 rows
```

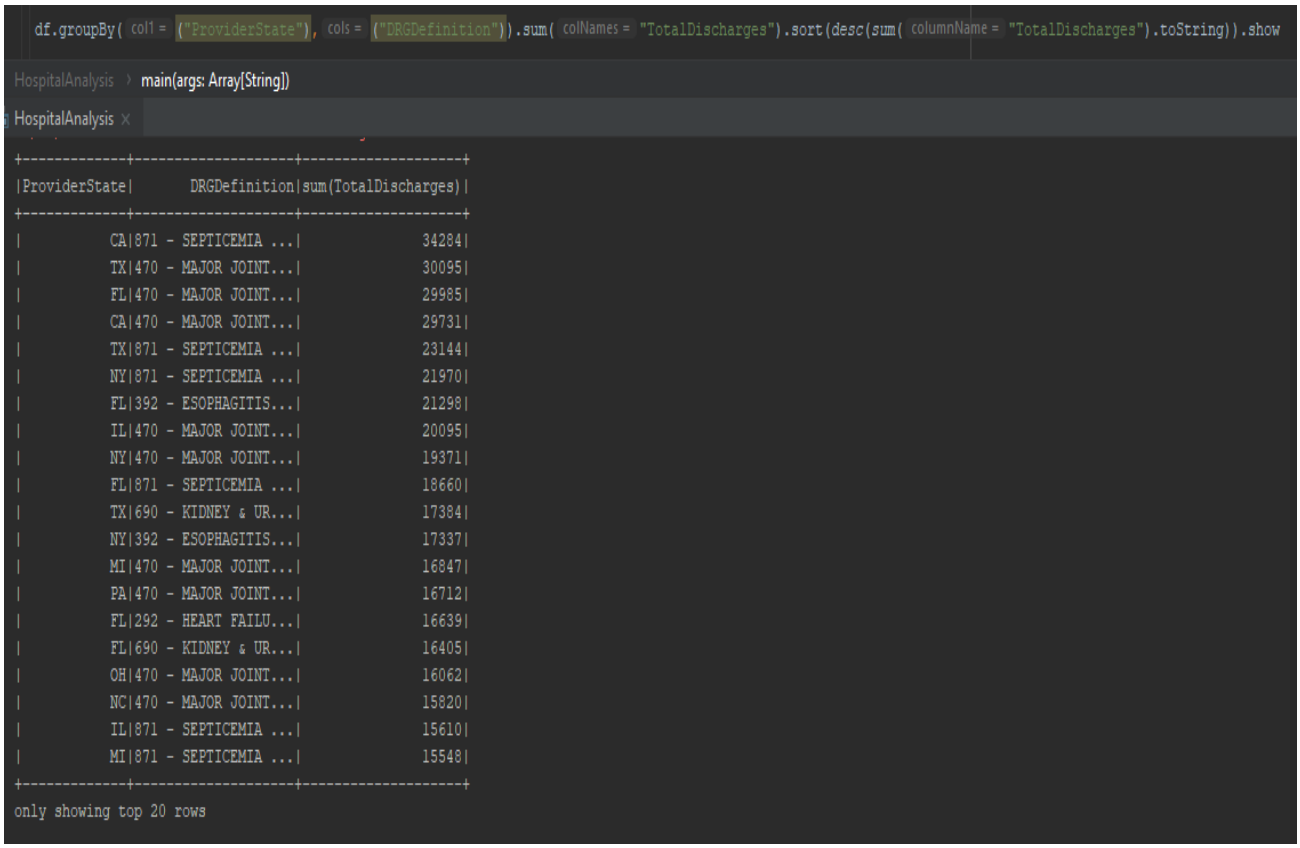
Problem Statement 4: Find out the total number of **Discharges** per state and for each disease.

df.groupBy(("ProviderState"),("DRGDefinition")).sum("TotalDischarges").show



Problem Statement 5: Sort the output in descending order of **totalDischarges**

df.groupBy(("ProviderState"),("DRGDefinition")).sum("TotalDischarges").sort(desc(sum("TotalDischarges").toString)).show



```
df.groupBy( col1 = ("ProviderState"), cols = ("DRGDefinition")).sum( colNames = "TotalDischarges").sort(desc(sum( columnName = "TotalDischarges").toString)).show
```

HospitalAnalysis › main(args: Array[String])

HospitalAnalysis ×

ProviderState	DRGDefinition	sum(TotalDischarges)
CA	871 - SEPTICEMIA ...	34284
TX	470 - MAJOR JOINT...	30095
FL	470 - MAJOR JOINT...	29985
CA	470 - MAJOR JOINT...	29731
TX	871 - SEPTICEMIA ...	23144
NY	871 - SEPTICEMIA ...	21970
FL	392 - ESOPHAGITIS...	21298
IL	470 - MAJOR JOINT...	20095
NY	470 - MAJOR JOINT...	19371
FL	871 - SEPTICEMIA ...	18660
TX	690 - KIDNEY & UR...	17384
NY	392 - ESOPHAGITIS...	17337
MI	470 - MAJOR JOINT...	16847
PA	470 - MAJOR JOINT...	16712
FL	292 - HEART FAILU...	16639
FL	690 - KIDNEY & UR...	16405
OH	470 - MAJOR JOINT...	16062
NC	470 - MAJOR JOINT...	15820
IL	871 - SEPTICEMIA ...	15610
MI	871 - SEPTICEMIA ...	15548

only showing top 20 rows