Case Study - IV

Hospital Case Study

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.

You can download the dataset used in this spark SQL use case from below link.

https://drive.google.com/open?id=13_YDmwENxOQI5asLRa6tOP8FgiqqM9jc

Objective – 1: Load file into spark.

Solution – 1: Prior to loading the file in Spark we create a Spark Session Object.

```
build.sbt ×   HospitalFinalTry.scala ×
1   import org.apache.spark.sql.SparkSession
2
3   object HospitalFinalTry {
4     def main(args: Array[String]): Unit = {
5       //println("Hello Hospital Use Case!")
6
7       val spark = SparkSession
8         .builder()
9         .master( master = "local")
10        .appName( name = "Spark SQL Use Case 1")
11        .config("spark.some.config.option", "some-value")
12        .getOrCreate()
13
14      println("Spark Session Object created")
15
```

Output:

```
HospitalFinalTry ×
18/05/25 19:18:18 INFO BlockManagerMasterEndpoint: Registering block ma
18/05/25 19:18:18 INFO BlockManagerMaster: Registered BlockManager Bloc
18/05/25 19:18:18 INFO BlockManager: Initialized BlockManager: BlockMan
18/05/25 19:18:18 INFO SharedState: Warehouse path is 'file:/C:/Users/A
Spark Session Object created
18/05/25 19:18:19 INFO MemoryStore: Block broadcast_0 stored as values
18/05/25 19:18:19 INFO MemoryStore: Block broadcast_0_piece0 stored as
18/05/25 19:18:19 INFO BlockManagerInfo: Added broadcast_0_piece0 in me
18/05/25 19:18:19 INFO SparkContext: Created broadcast 0 from csv at Ho
```

Now proceed to load the file into Spark and here we are loading into Data Frame.

```
16
17      val df1 = spark.sqlContext.read
18              .option("header", "true")
19              .option("inferSchema", "true")
20              .csv( path = "C:\\Users\\Ankith M\\Desktop\\Hadoop\\Spark\\Hospital Case Study\\inpatientCharges.csv")
21
22      println("Spark DF1 created!")
```

Output:

```
Run:  HospitalFinalTry ×
18/05/25 19:18:21 INFO BlockManagerInfo: Removed broadcast_4_piece0 on 1
18/05/25 19:18:21 INFO BlockManagerInfo: Removed broadcast_2_piece0 on 1
Spark DF1 created!
18/05/25 19:18:23 INFO FileSourceStrategy: Pruning directories with:
18/05/25 19:18:23 INFO FileSourceStrategy: Post-Scan Filters:
18/05/25 19:18:23 INFO FileSourceStrategy: Output Data Schema: struct<DR
```

Now create temporary view for the Dataframe.

```
25
26        df1.createOrReplaceTempView( viewName = "hospital_charges")
27
28        println("temporary view created!!!!")
29
```

Output:

```
HospitalFinalTry
18/05/25 19:18:23 INFO SparkSqlParser: Parsing command: hospital_charges
temporary view created!!!!
18/05/25 19:18:24 INFO FileSourceStrategy: Pruning directories with:
18/05/25 19:18:24 INFO FileSourceStrategy: Post-Scan Filters:
```

Objective – 2:

What is the average amount of AverageCoveredCharges per State.

```
30        // Objective 2 - average amount of AverageCoveredCharges per State.
31        df1.groupBy( col1 = "ProviderState").avg( colNames = "AverageCoveredCharges").show
32
```

Output:

```
HospitalFinalTry
18/05/25 19:28:58 INFO CodeGenerator: Code generated in 12.
+-------------+--------------------------+
18/05/25 19:28:58 INFO SparkContext: Invoking stop() from s
|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

18/05/25 19:28:58 INFO SparkUI: Stopped Spark web UI at http
```

Find out the AverageTotalPayments charges per state

```
// Objective 2.2 - find out the AverageTotalPayments charges per state.
df1.groupBy( col1 = "ProviderState").avg( colNames = "AverageTotalPayments").show
```

Output:

```
HospitalFinalTry
18/05/25 19:33:05 INFO DAGScheduler: Job 7 finished: show at HospitalFi
+-------------+------------------------+
|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

18/05/25 19:33:05 INFO CodeGenerator: Code generated in 11.895618 ms
```

Find out the AverageMedicarePayments charges per state.

```
// Objective 2.3 - find out the AverageMedicarePayments charges per state.
df1.groupBy( col1 = "ProviderState").avg( colNames = "AverageMedicarePayments").show
```

Output:

```
HospitalFinalTry
18/05/25 19:36:54 INFO DAGScheduler: Job 7 finished: show at Hospital
+-------------+---------------------------+
|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

18/05/25 19:36:54 INFO CodeGenerator: Code generated in 10.031903 ms
```

<u>Objective 3</u>

Find out the total number of Discharges per state and for each disease

```
// Objective 3.1 - Find out the total number of Discharges per state and for each disease
val res1 = df1.groupBy( col1 = "ProviderState",  cols = "DRGDefinition").sum( colNames = "TotalDischarges")
res1.show()
```

Output:

HospitalFinalTry

```
18/05/25 19:42:54 INFO SparkContext: Invoking stop() from shutdown
18/05/25 19:42:54 INFO SparkUI: Stopped Spark web UI at http://192.
+-------------+--------------------+--------------------+
|ProviderState|       DRGDefinition|sum(TotalDischarges)|
+-------------+--------------------+--------------------+
|           KY|065 - INTRACRANIA...|                1937|
|           NY|101 - SEIZURES W/...|                4503|
|           IN|149 - DYSEQUILIBRIUM|                 700|
|           IA|178 - RESPIRATORY...|                 540|
|           WI|202 - BRONCHITIS ...|                 338|
|           MO|208 - RESPIRATORY...|                1840|
|           WI|251 - PERC CARDIO...|                 417|
|           AR|281 - ACUTE MYOCA...|                 413|
|           AZ|292 - HEART FAILU...|                2643|
|           NY|292 - HEART FAILU...|               13289|
|           NV|293 - HEART FAILU...|                 519|
|           SD|303 - ATHEROSCLER...|                  53|
|           TN|305 - HYPERTENSIO...|                 730|
|           ME|308 - CARDIAC ARR...|                 312|
|           NV|372 - MAJOR GASTR...|                 126|
|           WA|392 - ESOPHAGITIS...|                3148|
|           WI|439 - DISORDERS O...|                 215|
|           MN|536 - FRACTURES O...|                 332|
|           DC|563 - FX, SPRN, S...|                  43|
|           CO|602 - CELLULITIS ...|                  86|
+-------------+--------------------+--------------------+
only showing top 20 rows

18/05/25 19:42:54 INFO MapOutputTrackerMasterEndpoint: MapOutputTra
18/05/25 19:42:54 INFO MemoryStore: MemoryStore cleared
18/05/25 19:42:54 INFO BlockManager: BlockManager stopped
```

Sort the output in descending order of totalDischarges

```
// Objective 3.1 - Find out the total number of Discharges per state and for each disease
val res1 = df1.groupBy( col1 = "ProviderState", cols = "DRGDefinition").sum( colNames = "TotalDischarges")
val res2 = res1.orderBy(org.apache.spark.sql.functions.col( colName = "sum(TotalDischarges)").desc)
res2.show()
```

Output:

```
HospitalFinalTry
18/05/25 19:47:11 INFO SparkUI: Stopped Spark web UI at http://192.16
+-------------+--------------------+--------------------+
|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

18/05/25 19:47:11 INFO MapOutputTrackerMasterEndpoint: MapOutputTracke
18/05/25 19:47:11 INFO MemoryStore: MemoryStore cleared
```

All the Objectives are solved and output is mentioned one by one. Below is the total code base used to complete the Hospital case study

```
import org.apache.spark.sql.SparkSession

object HospitalFinalTry {
  def main(args: Array[String]): Unit = {
    //println("Hello Hospital Use Case!")

    val spark = SparkSession
      .builder()
      .master("local")
      .appName("Spark SQL Use Case 1")
      .config("spark.some.config.option", "some-value")
```

```
      .getOrCreate()

    println("Spark Session Object created")

    // Objective 1 - Load the file.

    val df1 = spark.sqlContext.read
              .option("header", "true")
              .option("inferSchema", "true")
              .csv("C:\\Users\\Ankith M\\Desktop\\Hadoop\\Spark\\Hospital Case
Study\\inpatientCharges.csv")

    println("Spark DF1 created!")

    df1.show()

    df1.createOrReplaceTempView("hospital_charges")

    println("temporary view created!!!!")

    // Objective 2.1 - average amount of AverageCoveredCharges per State.
    df1.groupBy("ProviderState").avg("AverageCoveredCharges").show

    // Objective 2.2 - find out the AverageTotalPayments charges per state.
    df1.groupBy("ProviderState").avg("AverageTotalPayments").show

    // Objective 2.3 - find out the AverageMedicarePayments charges per state.
    df1.groupBy("ProviderState").avg("AverageMedicarePayments").show


    // Objective 3.1 - Find out the total number of Discharges per state and for
each disease
    val res1 = df1.groupBy("ProviderState", "DRGDefinition").sum("TotalDischarges")
    res1.show()


    // Objective 3.1 - Find out the total number of Discharges per state and for
each disease
    val res1 = df1.groupBy("ProviderState", "DRGDefinition").sum("TotalDischarges")
    val res2 =
res1.orderBy(org.apache.spark.sql.functions.col("sum(TotalDischarges)").desc)
    res2.show()


  }

}
```