DATA MANAGEMENT AND ORGANIZATION

Name : Sharon Zefanya Setiawan

NIM : 2501961022

Class : LA09

CODE

```
scala> spark.sql("SHOW DATABASES").show()
+------------+
|databaseName|
+------------+
|    bogus_db|
|     default|
|      emr_db|
|       gp_db|
|       hr_db|
|    sales_db|
|solutions_db|
|    store_db|
|    union_db|
+------------+
```

1. Find the top 15 most common first names of the company's male's employees. Create a report showing the first name, its frequency, and its rank.
   Answers:

```
scala> spark.sql("USE hr_db")
res5: org.apache.spark.sql.DataFrame = []

scala> spark.sql("SHOW TABLES FROM hr_db").show()
+--------+----------+-----------+
|database| tableName|isTemporary|
+--------+----------+-----------+
|   hr_db|hr_records|      false|
+--------+----------+-----------+
```

It can be proven that there is table `hr_records` in the database `hr_db`.

```
scala> val df = spark.sql("SELECT * from hr_records")
df: org.apache.spark.sql.DataFrame = [name_prefix: string, first_name: string ... 16 more fields]

scala> df.printSchema()
root
 |-- name_prefix: string (nullable = true)
 |-- first_name: string (nullable = true)
 |-- middle_initial: string (nullable = true)
 |-- last_name: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- date_of_birth: date (nullable = true)
 |-- date_of_joining: date (nullable = true)
 |-- salary: integer (nullable = true)
 |-- last_pct_hike: integer (nullable = true)
 |-- ssn: string (nullable = true)
 |-- phone_nbr: string (nullable = true)
 |-- place_name: string (nullable = true)
 |-- county: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- zip: string (nullable = true)
 |-- region: string (nullable = true)
 |-- user_name: string (nullable = true)
```

```
scala> df.show(15)
+-----------+----------+--------------+-----------+------+-------------+---------------+------+-------------+-----------+-----------+--------------+-----------+--------------+-----+-----+---------+-------------+
|name_prefix|first_name|middle_initial| last_name|gender|date_of_birth|date_of_joining|salary|last_pct_hike|        ssn|  phone_nbr|    place_name|     county|          city|state|  zip|   region|    user_name|
+-----------+----------+--------------+-----------+------+-------------+---------------+------+-------------+-----------+-----------+--------------+-----------+--------------+-----+-----+---------+-------------+
|        Ms.|   Hermila|             J|       Suhr|     F|   1992-09-04|     2017-09-09|168991|           12|275-17-8844|479-539-4593|  Peach Orchard|       Clay|  Peach Orchard|   AR|72453|    South|       hjsuhr|
|        Mr.|   Antonio|             Q|        Joy|     M|   1989-12-24|     2014-08-02| 53504|           30|646-23-6213|229-234-6154|     Rocky Ford|    Screven|     Rocky Ford|   GA|30455|    South|       aqjoy|
|      Prof.| Sebastian|             J|     Moores|     M|   1980-09-23|     2015-04-01|158859|            3|499-29-8139|272-231-9912|       Antwerp|    Jefferson|       Antwerp|   NY|13608|Northeast|     sjmoores|
|        Mr.|      Alec|             S|Rittenhouse|     M|   1974-06-01|     2001-03-19| 76105|            2|204-84-0943|229-873-6796|  Willedgeville| Willedgeville|  Willedgeville|   GA|31059|    South|asrittenhouse|
|        Mr.|    Reggie|             C|    Doughty|     M|   1966-11-27|     2013-01-04|134436|            8|321-11-0416|314-677-4501|    Springfield|     Greene|    Springfield|   MO|65809|  Midwest|     rcdoughty|
|      Prof.|    Elisha|             B|     Bottom|     M|   1972-09-23|     2018-05-22|119237|           22|517-49-8835|206-233-8897|        Dryden|     Chelan|        Dryden|   WA|98821|     West|     ebbottom|
|        Mr.|    Danilo|             R|      Irwin|     M|   1964-12-22|     1997-02-14|115449|           25|605-87-8120|319-326-7935|      Urbandale|       Polk|      Urbandale|   IA|50323|  Midwest|      drirwin|
|        Ms.|  Madalene|             D|     Dierks|     F|   1957-10-29|     2014-03-21|104968|            6|050-02-8165|503-778-8049|     Deer Island|   Columbia|     Deer Island|   OR|97054|     West|     mddierks|
|       Drs.|   Micheal|             W|  Campanella|     F|   1969-11-15|     2004-09-13| 71948|            7|204-84-1953|216-821-3070|   Leavittsburg|   Trumbull|   Leavittsburg|   OH|44430|  Midwest| mwcampanella|
|       Mrs.|     Wilma|             V|      Trail|     F|   1966-11-28|     2001-07-19|114397|           23|405-73-1471|505-988-5623|         Organ|     Do?a Ana|         Organ|   NM|88052|     West|      wvtrail|
|      Prof.|     Jorge|             K|    Mohamed|     M|   1970-06-23|     2016-10-31|106883|            2|486-29-1020|262-939-1044|       Houlton|  St. Croix|       Houlton|   WI|54082|  Midwest|     jkmohamed|
|        Mr.|     Isaac|             G|  Cumberland|     M|   1982-09-15|     2008-05-04| 79923|           14|059-02-6964|209-304-3773|       El Monte| Los Angeles|       El Monte|   CA|91734|     West|  igcumberland|
|        Mr.|   Emerson|             E|     Rayner|     M|   1972-02-22|     1996-05-24|179309|           16|376-37-5492|316-451-6275|Shawnee Mission|    Johnson|Shawnee Mission|   KS|66218|  Midwest|     eerayner|
|        Mr.|    Booker|             G|       Olmo|     M|   1980-05-17|     2009-11-17| 55634|            7|567-99-2442|236-384-0434|  Edwardsville|Northumberland|  Edwardsville|   VA|22456|    South|      bgolmo|
|      Prof.|  Margarito|             B|    Cushman|     M|   1990-01-07|     2017-05-22|173211|           30|142-23-1764|503-264-1465|       Rickreall|       Polk|       Rickreall|   OR|97371|     West|     mbcushman|
+-----------+----------+--------------+-----------+------+-------------+---------------+------+-------------+-----------+-----------+--------------+-----------+--------------+-----+-----+---------+-------------+
only showing top 15 rows
```

`hr_records` includes 18 attributes and primarily contains string data. It is clear from the above scheme that the names are already divided into first, middle, and last names, negating the necessity for initial separation.

```
scala> val top15_male = spark.sql("SELECT first_name, COUNT(first_name) as frequency, DENSE_RANK() OVER(ORDER BY COUNT(first_name) DESC) as rank FROM hr_records WHERE g
ender = 'M' GROUP BY first_name ORDER BY rank")
top15_male: org.apache.spark.sql.DataFrame = [first_name: string, frequency: bigint ... 1 more field]
```

```
scala> top15_male.show(15, false)
+----------+---------+----+
|first_name|frequency|rank|
+----------+---------+----+
|Fidel     |253      |1   |
|Ralph     |252      |2   |
|Otis      |250      |3   |
|Terrance  |250      |3   |
|Otha      |249      |4   |
|Lamar     |247      |5   |
|Efrain    |246      |6   |
|Alvaro    |244      |7   |
|Phil      |243      |8   |
|Walker    |243      |8   |
|Keith     |242      |9   |
|Myron     |242      |9   |
|Amos      |242      |9   |
|Luigi     |242      |9   |
|Garfield  |241      |10  |
+----------+---------+----+
only showing top 15 rows
```

The output above lists the 15 male employees with the most popular first names. Fidel is the most common first name, occurring in 253 people. We can infer from the output that 241-253 male employees share the same first name.

```
scala> top15_male.createOrReplaceTempView("top15_male")

scala> spark.sql("SELECT COUNT(*) as count FROM top15_male").show(false)
+-----+
|count|
+-----+
|1219 |
+-----+
```

It can be proven that the total records of final output is 1219.

```
scala> top15_male.write.format("csv").
     | mode("overwrite").
     | option("sep", "-").
     | option("header", "true").
     | save("/user/2501961022/solution")
```

Data saved in the hdfs directory: user/2501961022/solution as csv files.

```
scala> val df = spark.read.format("csv").
     | option("sep", "-").
     | option("header", "true").
     | load("/user/2501961022/solution")
df: org.apache.spark.sql.DataFrame = [first_name: string, frequency: string ... 1 more field]
```

```
scala> df.show(false)
+----------+---------+----+
|first_name|frequency|rank|
+----------+---------+----+
|Fidel     |253      |1   |
|Ralph     |252      |2   |
|Otis      |250      |3   |
|Terrance  |250      |3   |
|Otha      |249      |4   |
|Lamar     |247      |5   |
|Efrain    |246      |6   |
|Alvaro    |244      |7   |
|Phil      |243      |8   |
|Walker    |243      |8   |
|Keith     |242      |9   |
|Amos      |242      |9   |
|Myron     |242      |9   |
|Luigi     |242      |9   |
|Garfield  |241      |10  |
|Frank     |240      |11  |
|Roderick  |239      |12  |
|Geraldo   |239      |12  |
|Roger     |238      |13  |
|Darell    |238      |13  |
+----------+---------+----+
only showing top 20 rows
```

Data read from directory successfully.

2. Use the EMR data to find the total number of emergency department visits that were due to influenza-like illness and/or pneumonia that resulted in hospitalization for the months of May, June & July.

Answers:

```
scala> val emr_data = spark.read.format("parquet").load("/user/verulam_blue/data/emr_data")
emr_data: org.apache.spark.sql.DataFrame = [extract_date: date, date_of_visit: date ... 4 more fields]

scala> emr_data.createOrReplaceTempView("emr_data")
```

```
scala> emr_data.show(false)
+------------+------------+--------+--------------+-------------+-----------------+
|extract_date|date_of_visit|mod_zcta|total_ed_visits|ili_pne_visits|ili_pne_admissions|
+------------+------------+--------+--------------+-------------+-----------------+
|2020-07-25  |2020-07-14  |11420   |40            |1            |1                |
|2020-07-25  |2020-07-02  |10038   |27            |0            |0                |
|2020-07-24  |2020-05-31  |10019   |33            |0            |0                |
|2020-07-25  |2020-03-01  |11210   |70            |4            |1                |
|2020-07-25  |2020-06-28  |11223   |43            |0            |0                |
|2020-07-24  |2020-03-12  |11417   |31            |9            |1                |
|2020-07-25  |2020-06-04  |11225   |29            |0            |0                |
|2020-07-24  |2020-03-16  |10463   |63            |13           |4                |
|2020-07-25  |2020-04-06  |11220   |68            |19           |6                |
|2020-07-25  |2020-04-22  |10038   |13            |0            |0                |
|2020-07-24  |2020-06-06  |10044   |9             |1            |1                |
|2020-07-25  |2020-05-19  |10024   |18            |0            |0                |
|2020-07-25  |2020-03-31  |10458   |102           |23           |8                |
|2020-07-24  |2020-05-30  |11379   |12            |0            |0                |
|2020-07-24  |2020-05-09  |11433   |23            |1            |1                |
|2020-07-24  |2020-07-06  |11692   |19            |0            |0                |
|2020-07-25  |2020-03-27  |10474   |24            |5            |2                |
|2020-07-24  |2020-07-18  |10475   |24            |0            |0                |
|2020-07-25  |2020-04-08  |10471   |13            |5            |1                |
|2020-07-25  |2020-05-06  |11434   |47            |3            |3                |
+------------+------------+--------+--------------+-------------+-----------------+
only showing top 20 rows
```

```
scala> emr_data.printSchema()
root
 |-- extract_date: date (nullable = true)
 |-- date_of_visit: date (nullable = true)
 |-- mod_zcta: string (nullable = true)
 |-- total_ed_visits: integer (nullable = true)
 |-- ili_pne_visits: integer (nullable = true)
 |-- ili_pne_admissions: integer (nullable = true)
```

The `emr_data` table contains 6 attributes. The visit date in date format, the total number of trips to the emergency room due to influenza-like illness and/or pneumonia, and the number of patients with influenza-like/pneumonia illnesses (which can be determined from the number of admissions) are all known from the table.

BASED ON QUESTIONS:

```
scala> val ili_pne_adm = spark.sql("""SELECT date_of_visit, ili_pne_visits, ili_pne_admissions FROM emr_data WHERE
     | (MONTH(date_of_visit) BETWEEN 5 AND 7) AND
     | ili_pne_admissions > 0""")
ili_pne_adm: org.apache.spark.sql.DataFrame = [date_of_visit: date, ili_pne_visits: int ... 1 more field]
```

```
scala> ili_pne_adm.show(false)
+-------------+--------------+-----------------+
|date_of_visit|ili_pne_visits|ili_pne_admissions|
+-------------+--------------+-----------------+
|2020-07-14   |1             |1                |
|2020-06-06   |1             |1                |
|2020-05-09   |1             |1                |
|2020-05-06   |3             |3                |
|2020-06-24   |1             |1                |
|2020-06-16   |1             |1                |
|2020-05-02   |2             |1                |
|2020-06-02   |4             |2                |
|2020-05-16   |1             |1                |
|2020-05-10   |2             |1                |
|2020-05-11   |1             |1                |
|2020-05-01   |8             |1                |
|2020-07-05   |2             |1                |
|2020-07-04   |1             |1                |
|2020-05-02   |7             |2                |
|2020-07-15   |1             |1                |
|2020-07-21   |2             |1                |
|2020-06-09   |2             |1                |
|2020-06-14   |3             |1                |
|2020-05-29   |1             |1                |
+-------------+--------------+-----------------+
only showing top 20 rows
```

```
scala> ili_pne_adm.createOrReplaceTempView("ili_pne_adm")

scala> val ili_pne_vis = spark.sql("""SELECT MONTH(date_of_visit) as month, SUM(ili_pne_visits) as total_visits
     | FROM ili_pne_adm WHERE(MONTH(date_of_visit) BETWEEN 5 AND 7) GROUP BY month ORDER BY month""")
ili_pne_vis: org.apache.spark.sql.DataFrame = [month: int, total_visits: bigint]
```

```
scala> ili_pne_vis.show(false)
+-----+------------+
|month|total_visits|
+-----+------------+
|5    |382753      |
|6    |192251      |
|7    |216901      |
+-----+------------+
```

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

scala> val df = ili_pne_vis.withColumn(
     | "month",
     | when(col("month") === "5", "May").
     | when(col("month") === "6", "June").
     | when(col("month") === "7", "July").
     | otherwise(col("month")))
df: org.apache.spark.sql.DataFrame = [month: string, total_visits: bigint]

scala> df.show(false)
+-----+------------+
|month|total_visits|
+-----+------------+
|May  |382753      |
|June |192251      |
|July |216901      |
+-----+------------+
```

As a result, the statistics above represent the total number of emergency room visits for pneumonia and/or influenza-like illness that required hospitalization in May, June, and July. 382753 visitors, or the most, were recorded in the month of May.

ACCORDING TO OUTPUT REQUIREMENTS AND SAMPLE RESULTS (FINAL ANSWER):

```
scala> val total_ed = spark.sql("""SELECT MONTH(date_of_visit) as month, SUM(ili_pne_admissions) as total
     | FROM emr_data WHERE(MONTH(date_of_visit) BETWEEN 5 AND 7) GROUP BY month ORDER BY month""")
total_ed: org.apache.spark.sql.DataFrame = [month: int, total: bigint]

scala> total_ed.show(false)
+-----+------+
|month|total |
+-----+------+
|5    |200801|
|6    |113289|
|7    |122021|
+-----+------+
```

According to the data above, May had the highest number of visits with 200801.

```
scala> total_ed.write.format("csv").
     | mode("overwrite").
     | option("sep", "-").
     | option("header", "true").
     | save("/user/2501961022/solution")
```

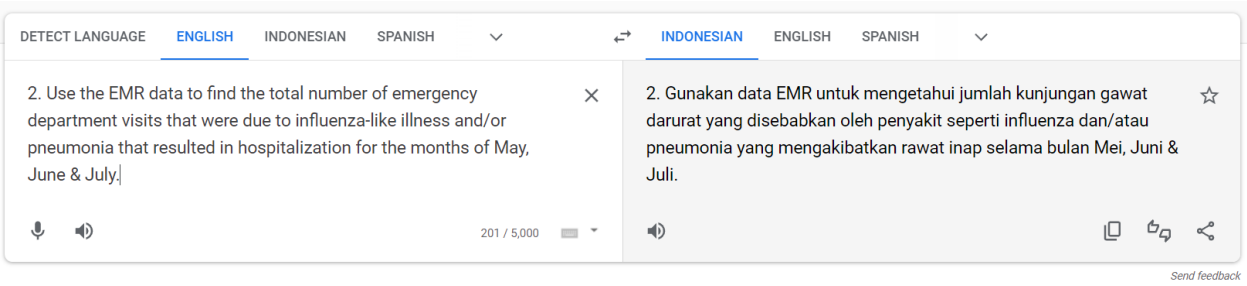
Data saved in the hdfs directory: user/2501961022/solution as csv files.

```
scala> val df = spark.read.format("csv").
     | option("sep", "-").
     | option("header", "true").
     | load("/user/2501961022/solution")
df: org.apache.spark.sql.DataFrame = [month: string, total: string]

scala> df.show(false)
+-----+------+
|month|total |
+-----+------+
|5    |200801|
|6    |113289|
|7    |122021|
+-----+------+
```

Data read from directory successfully.

NOTES



Menurut soal yang diberikan, kita disuruh untuk menampilkan output berupa total visit yang disebabkan oleh penyakit seperti influenza/pneumonia (yang dirawat inap) selama bulan Mei, Juni, & Juli. Seperti yang kita ketahui, admission merupakan tempat pasien dalam mendaftarkan serta membayar total biaya rawat inap mereka. Jadi, kolom ili_pne_admissions di atas dapat diartikan sebagai total pasien yang melakukan rawat inap yang disebabkan oleh penyakit seperti influenza/pneumonia. Inti: visitor tidak perlu ke admission jika ingin mengunjungi pasien. Berdasarkan yang diminta di soal, kita diminta untuk menampilkan jumlah kunjungan (ili_pne_visits) khusus untuk pasien yang dirawat inap. Oleh karena itu, saya berusaha memberikan output sesuai yang diminta oleh soal tertulis (total pengunjung yang mengunjungi rawat inap).

 # OUTPUT REQUIREMENTS:

- Place the result data in the hdfs directory: /user/studentID/solution
- Results should show month of visit and number of hospitalizations.
- Use a text format with a tab as the columnar delimiter.

 # RESULTS:

```
+---+----------+
|5  |200801|
|6  |113289|
|7  |122021|
+---+----------+
```

Namun, contoh output yang tertera pada soal menunjukkan hasil yang berbeda yaitu total pasien yang dirawat inap pada bulan Mei, Juni, dan Juli. Oleh karena itu, saya memberikan 2 contoh jawaban, dengan tetap melakukan penyimpanan pada hdfs berupa contoh output yang ada di lembar soal → **Final Answer** 😊

3. Working with the "gp_db" database use the "items" column from the metastore table "gp_rx" together with the "gp_address" table to find the total number of prescriptions made by each GP practice in the city of Bolton.
    - The "items" column, in the metastore table "gp_rx", represents the "total number of items prescribed".
    - The first 4 letters of postcodes for GP practices in Bolton are: 'BL1 ','BL2 ','BL3 '

    Answers:

```
scala> spark.sql("USE gp_db")
res11: org.apache.spark.sql.DataFrame = []

scala> spark.sql("SHOW TABLES FROM gp_db").show()
+--------+-------------------+-----------+
|database|          tableName|isTemporary|
+--------+-------------------+-----------+
|   gp_db|         gp_address|      false|
|   gp_db|              gp_rx|      false|
|   gp_db|practice_demograp...|      false|
|        |            emr_data|       true|
|        |        ili_pne_adm|       true|
+--------+-------------------+-----------+
```

    It can be proven that there are table `gp_rx and gp_address` in the database gp_db.

```
scala> val gp_address = spark.sql("SELECT * FROM gp_address")
gp_address: org.apache.spark.sql.DataFrame = [date: string, practice_code: string ... 6 more fields]

scala> gp_address.printSchema()
root
 |-- date: string (nullable = true)
 |-- practice_code: string (nullable = true)
 |-- surgery_name: string (nullable = true)
 |-- address_1: string (nullable = true)
 |-- address_2: string (nullable = true)
 |-- address_3: string (nullable = true)
 |-- address_4: string (nullable = true)
 |-- postcode: string (nullable = true)
```

```
scala> gp_address.show(false)
+------+-------------+-----------------------------+-----------------------+-----------------------+---------------+---------+--------+
|date  |practice_code|surgery_name                 |address_1              |address_2              |address_3      |address_4|postcode|
+------+-------------+-----------------------------+-----------------------+-----------------------+---------------+---------+--------+
|201512|A81001       |THE DENSHAM SURGERY          |THE HEALTH CENTRE      |LAWSON STREET          |STOCKTON ON TEES|CLEVELAND|TS18 1HU|
|201512|A81002       |QUEENS PARK MEDICAL CENTRE   |QUEENS PARK MEDICAL CTR|FARRER STREET          |STOCKTON ON TEES|CLEVELAND|TS18 2AW|
|201512|A81003       |VICTORIA MEDICAL PRACTICE    |THE HEALTH CENTRE      |VICTORIA ROAD          |HARTLEPOOL     |CLEVELAND|TS26 8DB|
|201512|A81004       |WOODLANDS ROAD SURGERY       |6 WOODLANDS ROAD       |                       |MIDDLESBROUGH  |CLEVELAND|TS1 3BE |
|201512|A81005       |SPRINGWOOD SURGERY           |SPRINGWOOD SURGERY     |RECTORY LANE           |GUISBOROUGH    |         |TS14 7DJ|
|201512|A81006       |TENNANT STREET MEDICAL PRACTICE|TENNANT ST MED PRACT |FARRER STREET          |STOCKTON ON TEES|CLEVELAND|TS18 2AT|
|201512|A81007       |BANKHOUSE SURGERY            |ONE LIFE HARTLEPOOL    |PARK ROAD              |HARTLEPOOL     |CLEVELAND|TS24 7PW|
|201512|A81008       |ALBERT HOUSE CLINIC          |LOW GRANGE HEALTH VILLAGE|NORMANBY ROAD        |MIDDLESBROUGH  |CLEVELAND|TS6 6TD |
|201512|A81009       |VILLAGE MEDICAL CENTRE       |THE VILLAGE MEDICAL CTR|400/404 LINTHORPE ROAD |MIDDLESBROUGH  |CLEVELAND|TS5 6HF |
|201512|A81011       |CHADWICK PRACTICE            |ONE LIFE HARTLEPOOL    |PARK ROAD              |HARTLEPOOL     |CLEVELAND|TS24 7PW|
|201512|A81012       |WESTBOURNE MEDICAL CENTRE    |WESTBOURNE MEDICAL CENTRE|7 TRINITY MEWS N. ORMESBY|MIDDLESBROUGH|         |TS3 6AL |
|201512|A81013       |BROTTON SURGERY              |BROTTON SURGERY        |ALFORD ROAD BROTTON    |SALTBURN BY SEA|         |TS12 2FF|
|201512|A81014       |QUEENSTREE PRACTICE          |THE HEALTH CENTRE      |QUEENSWAY              |BILLINGHAM     |CLEVELAND|TS23 2LA|
|201512|A81015       |LAGAN SURGERY                |LAGAN SURGERY          |20 KIRKLEATHAM STREET  |REDCAR         |CLEVELAND|TS10 1TZ|
|201512|A81016       |PARK SURGERY                 |PARK SURGERY  ONE LIFE |LINTHORPE ROAD         |MIDDLESBROUGH  |CLEVELAND|TS1 3QY |
|201512|A81017       |WOODBRIDGE PRACTICE          |THE HEALTH CENTRE      |TRENCHARD AVE  THORNABY|STOCKTON ON TEES|CLEVELAND|TS17 0EE|
|201512|A81018       |BENTLEY MEDICAL PRACTICE     |REDCAR PRIMARY CARE HOSP|WEST DYKE ROAD        |REDCAR         |CLEVELAND|TS10 4NW|
|201512|A81019       |CROSSFELL HEALTH CENTRE      |CROSSFELL ROAD         |BERWICK HILLS          |MIDDLESBROUGH  |CLEVELAND|TS3 7RL |
|201512|A81020       |MARTONSIDE MEDICAL CENTRE    |MARTONSIDE MEDICAL CENTRE|1A MARTONSIDE WAY    |MIDDLESBROUGH  |CLEVELAND|TS4 3BU |
|201512|A81021       |NORMANBY MEDICAL CENTRE      |NORMANBY MEDICAL CENTRE|LOW GRANGE HV NORMANBY RD|MIDDLESBROUGH|CLEVELAND|TS6 6TD |
+------+-------------+-----------------------------+-----------------------+-----------------------+---------------+---------+--------+
only showing top 20 rows
```

```
scala> val gp_rx = spark.sql("SELECT * FROM gp_rx")
gp_rx: org.apache.spark.sql.DataFrame = [sha: string, pct: string ... 8 more fields]

scala> gp_rx.printSchema()
root
 |-- sha: string (nullable = true)
 |-- pct: string (nullable = true)
 |-- practice_code: string (nullable = true)
 |-- bnf_code: string (nullable = true)
 |-- bnf_name: string (nullable = true)
 |-- items: integer (nullable = true)
 |-- nic: float (nullable = true)
 |-- act_cost: float (nullable = true)
 |-- quantity: integer (nullable = true)
 |-- period: string (nullable = true)
```

```
scala> gp_rx.show(false)
+---+---+------------+--------------+--------------------------+-----+-----+--------+--------+------+
|sha|pct|practice_code|bnf_code      |bnf_name                  |items|nic  |act_cost|quantity|period|
+---+---+------------+--------------+--------------------------+-----+-----+--------+--------+------+
|Q44|RJN|Y05218      |0501013K0AAAJAJ|Co-Amoxiclav_Tab 500mg/125mg|1    |3.59 |3.33    |21      |201512|
|Q44|RJN|Y05218      |0501130R0AAAAAA|Nitrofurantoin_Cap 50mg     |1    |14.39|13.42   |28      |201512|
|Q44|RTV|Y04937      |0401020K0AAAHAH|Diazepam_Tab 2mg            |1    |0.51 |0.58    |14      |201512|
|Q44|RTV|Y04937      |0401020P0AAABAB|Lorazepam_Tab 1mg           |1    |2.65 |2.46    |28      |201512|
|Q44|RTV|Y04937      |0402010ABAAABAB|Quetiapine_Tab 25mg         |2    |2.01 |2.08    |84      |201512|
|Q44|RTV|Y04937      |0402010ABAAACAC|Quetiapine_Tab 100mg        |1    |0.56 |0.63    |14      |201512|
|Q44|RTV|Y04937      |0402010ABAAAVAV|Quetiapine_Tab 50mg M/R     |1    |7.89 |7.41    |7       |201512|
|Q44|RTV|Y04937      |0402010ABAAAXAX|Quetiapine_Tab 300mg M/R    |1    |79.33|73.46   |28      |201512|
|Q44|RTV|Y04937      |0402010ABAAAYAY|Quetiapine_Tab 400mg M/R    |1    |26.39|24.51   |7       |201512|
|Q44|RTV|Y04937      |0402010D0AAAJAJ|Chlorpromazine HCl_Tab 50mg |1    |0.57 |0.64    |7       |201512|
|Q44|RTV|Y04937      |0402010D0AAAKAK|Chlorpromazine HCl_Tab 100mg|1    |2.35 |2.19    |28      |201512|
|Q44|RTV|Y04937      |040201060AAACAC|Olanzapine_Tab 10mg         |1    |1.44 |1.34    |28      |201512|
|Q44|RTV|Y04937      |040201060AAADAD|Olanzapine_Tab 2.5mg        |2    |1.98 |1.86    |56      |201512|
|Q44|RTV|Y04937      |0403010B0AAAGAG|Amitriptyline HCl_Tab 10mg  |1    |1.43 |1.43    |42      |201512|
|Q44|RTV|Y04937      |0403010B0AAAHAH|Amitriptyline HCl_Tab 25mg  |2    |1.96 |1.84    |56      |201512|
|Q44|RTV|Y04937      |0403010B0AAAIAI|Amitriptyline HCl_Tab 50mg  |1    |1.18 |1.1     |28      |201512|
|Q44|RTV|Y04937      |0403010X0AAABAB|Trazodone HCl_Cap 100mg     |1    |21.11|19.63   |42      |201512|
|Q44|RTV|Y04937      |0403030Q0AAAAAA|Sertraline HCl_Tab 50mg     |3    |2.76 |2.79    |42      |201512|
|Q44|RTV|Y04937      |0403030Q0AAABAB|Sertraline HCl_Tab 100mg    |2    |1.33 |1.46    |21      |201512|
|Q44|RTV|Y04937      |0403040W0AAAAAA|Venlafaxine_Tab 37.5mg      |1    |1.18 |1.2     |28      |201512|
+---+---+------------+--------------+--------------------------+-----+-----+--------+--------+------+
only showing top 20 rows
```

From the data above, it can be concluded that we need to join the gp_address and gp_rx tables with the practice_code attribute as the key.

```
scala> val gp_bolton = spark.sql("""SELECT a.practice_code, a.surgery_name, SUM(b.items) as total_prescriptions
     | FROM gp_address a JOIN gp_rx b
     | ON a.practice_code = b.practice_code
     | WHERE a.postcode LIKE 'BL1%'
     | OR a.postcode LIKE 'BL2%'
     | OR a.postcode LIKE 'BL3%'
     | GROUP BY a.practice_code, a.surgery_name
     | ORDER BY a.practice_code DESC""")
gp_bolton: org.apache.spark.sql.DataFrame = [practice_code: string, surgery_name: string ... 1 more field]
```

```
scala> gp_bolton.show(false)
+------------+-----------------------------------+------------------+
|practice_code|surgery_name                       |total_prescriptions|
+------------+-----------------------------------+------------------+
|Y04600       |BARDOC GP OOH                      |3042              |
|Y03641       |BOLTON COMMUNITY DRUG AND ALCOHOL SERV|2267           |
|Y03366       |OLIVE FAMILY PRACTICE              |5030              |
|Y03364       |GREAT LEVER PRACTICE               |5619              |
|Y03079       |BOLTON COMMUNITY PRACTICE          |22209             |
|Y02943       |NEUROLOGY LONG TERM CONDITIONS     |56                |
|Y02790       |BOLTON MEDICAL CENTRE              |4155              |
|Y02319       |BOLTON GENERAL PRACTICE            |5095              |
|Y00747       |HALLIWELL HEALTH & CHILDREN'S CENTRE|165              |
|Y00552       |MINOR TREATMENT CLINIC             |1                 |
|Y00448       |DIABETES CENTRE                    |150               |
|Y00233       |THE PARALLEL                       |1                 |
|Y00215       |ORTHOPAEDIC & RHEUMATOLOGY         |70                |
|Y00208       |TIER 2 DERMATOLOGY SERVICE         |9                 |
|Y00186       |3D MEDICAL CENTRE                  |1547              |
|P82661       |INTERMEDIATE CARE                  |470               |
|P82660       |DEANE CLINIC 1                     |6620              |
|P82654       |BOLTON HOSPICE                     |14                |
|P82640       |AL FAL MEDICAL GROUP               |6785              |
|P82634       |WYRESDALE ROAD SURGERY             |5443              |
+------------+-----------------------------------+------------------+
only showing top 20 rows
```

The table above shows the total prescriptions for each GP in Bolton.

```
scala> gp_bolton.write.format("json").
     | mode("overwrite").
     | option("header", "true").
     | save("/user/2501961022/solution")
```

Data saved in the hdfs directory: user/2501961022/solution as csv files.

```
scala> val df = spark.read.format("json").
     | option("header", "true").
     | load("/user/2501961022/solution")
df: org.apache.spark.sql.DataFrame = [practice_code: string, surgery_name: string ... 1 more field]
```

```
scala> df.show(false)
+-------------+-----------------------------------+------------------+
|practice_code|surgery_name                       |total_prescriptions|
+-------------+-----------------------------------+------------------+
|Y03641       |BOLTON COMMUNITY DRUG AND ALCOHOL SERV|2267              |
|Y00747       |HALLIWELL HEALTH & CHILDREN'S CENTRE  |165               |
|P82018       |THE ALASTAIR ROSS MEDICAL PRACTICE    |9081              |
|P82607       |WALMSLEY-CROMPTON HEALTH CENTRE       |9392              |
|Y02943       |NEUROLOGY LONG TERM CONDITIONS        |56                |
|P82036       |LITTLE LEVER HEALTH CENTRE 2          |6191              |
|P82021       |KIRBY-CROMPTON HEALTH CENTRE          |8710              |
|P82020       |LITTLE LEVER HEALTH CENTRE 1          |6896              |
|P82633       |GREAT LEVER HEALTH CENTRE 1           |5360              |
|P82624       |ORIENT HOUSE MEDICAL CENTRE           |6108              |
|P82613       |SPRING VIEW MEDICAL CENTRE            |10541             |
|Y03079       |BOLTON COMMUNITY PRACTICE             |22209             |
|P82004       |SWAN LANE MEDICAL CENTRE              |15775             |
|Y00215       |ORTHOPAEDIC & RHEUMATOLOGY            |70                |
|P82625       |CHARLOTTE STREET SURGERY              |2618              |
|P82023       |MANDALAY MEDICAL CENTRE               |12220             |
|P82011       |TONGE FOLD HEALTH CENTRE              |9111              |
|P82009       |ST HELENS ROAD PRACTICE               |14147             |
|P82001       |THE DUNSTAN PARTNERSHIP               |18419             |
|Y02319       |BOLTON GENERAL PRACTICE               |5095              |
+-------------+-----------------------------------+------------------+
only showing top 20 rows
```

Data read from directory successfully.