Nama : Raina Imtiyaz

NIM : 2502010976

Class : LB09

Video Link: https://binusianorg-my.sharepoint.com/personal/raina_imtiyaz_binus_ac_id/_layouts/15/guestaccess.aspx?docid=1499e510e3abf449ca34d9a19f2eff864&authkey=AaaHXESTUHVa1eN9n4Scrxw&e=QgdaJA

Hyperlink: 2502010976_Raina Imtiyaz.mp4

Codes

1.

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

```
scala> spark.sql("use hr_db")
res1: org.apache.spark.sql.DataFrame = []

scala> spark.sql("show tables from hr_db").show()
+--------+----------+-----------+
|database| tableName|isTemporary|
+--------+----------+-----------+
|   hr_db|hr_records|      false|
+--------+----------+-----------+


scala> spark.sql("desc hr_records").show()
+---------------+---------+-------+
|       col_name|data_type|comment|
+---------------+---------+-------+
|    name_prefix|   string|   null|
|     first_name|   string|   null|
| middle_initial|   string|   null|
|      last_name|   string|   null|
|         gender|   string|   null|
|  date_of_birth|     date|   null|
|date_of_joining|     date|   null|
|         salary|      int|   null|
|  last_pct_hike|      int|   null|
|            ssn|   string|   null|
|      phone_nbr|   string|   null|
|     place_name|   string|   null|
|         county|   string|   null|
|           city|   string|   null|
|          state|   string|   null|
|            zip|   string|   null|
|         region|   string|   null|
|      user_name|   string|   null|
+---------------+---------+-------+
```

```
scala> spark.sql("select * from hr_records").show(10)
+-----------+----------+--------------+-----------+------+-------------+---------------+------+-------------+-----------+-----------+-------------+-------------+-------------+-----+-----
|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|212-231-9912|      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|Milledgeville|Milledgeville|Milledgeville|   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|
+-----------+----------+--------------+-----------+------+-------------+---------------+------+-------------+-----------+-----------+-------------+-------------+-------------+-----+-----
+---------+-------------+
only showing top 10 rows
```

```
scala> val hr_rec1 = 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 gender LIKE 'M'
 GROUP BY first_name ORDER BY rank ASC")
hr_rec1: org.apache.spark.sql.DataFrame = [first_name: string, frequency: bigint ... 1 more field]

scala> hr_rec1.show(15)
+----------+---------+----+
|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
```

```
scala> hr_rec1.write.format("csv").mode("overwrite").option("header", "true").option("delimiter","-").save("hdfs:///user/2502010976/solution")

scala> val hr_test = spark.read.format("csv").option("header","true").load("hdfs:///user/2502010976/solution/")
hr_test: org.apache.spark.sql.DataFrame = [first_name-frequency-rank: string]

scala> hr_test.show(15)
+-------------------------+
|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|
+-------------------------+
only showing top 15 rows
```

2.

```
scala> spark.sql("use emr_db")
res1: org.apache.spark.sql.DataFrame = []

scala> spark.sql("show tables from emr_db").show()
+--------+---------+-----------+
|database|tableName|isTemporary|
+--------+---------+-----------+
|  emr_db| emr_data|      false|
+--------+---------+-----------+


scala> spark.sql("desc emr_data").show()
+-------------------+---------+-------+
|           col_name|data_type|comment|
+-------------------+---------+-------+
|       extract_date|     date|   null|
|      date_of_visit|     date|   null|
|           mod_zcta|   string|   null|
|    total_ed_visits|      int|   null|
|     ili_pne_visits|      int|   null|
|ili_pne_admissions|      int|   null|
+-------------------+---------+-------+
```

```
scala> spark.sql("select * from emr_data").show(10)
+------------+-------------+--------+---------------+--------------+------------------+
|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|
+------------+-------------+--------+---------------+--------------+------------------+
only showing top 10 rows
```

```
scala> val emr_dat1 = spark.sql("select MONTH(date_of_visit) AS month_of_visit, SUM(ili_pne_admissions) AS number_of_hospitalizations FROM emr_data WHERE (MONTH(date_of_visit)
BETWEEN '5' AND '7') GROUP BY MONTH(date_of_visit) ORDER BY month_of_visit ASC")
emr_dat1: org.apache.spark.sql.DataFrame = [month_of_visit: int, number_of_hospitalizations: bigint]

scala> emr_dat1.show()
+-------------+--------------------------+
|month_of_visit|number_of_hospitalizations|
+-------------+--------------------------+
|            5|                    200801|
|            6|                    113289|
|            7|                    122021|
+-------------+--------------------------+
```

```
scala> emr_dat1.write.format("csv").mode("overwrite").option("header","true").option("delimiter","\t").save("hdfs:///user/2502010976/solution")

scala> val emr_test = spark.read.format("csv").option("header", "true").load("hdfs:///user/2502010976/solution/")
emr_test: org.apache.spark.sql.DataFrame = [month_of_visit      number_of_hospitalizations: string]

scala> emr_test.show()
+----------------------------------------+
|month_of_visit number_of_hospitalizations|
+----------------------------------------+
|                                5     200801|
|                                6     113289|
|                                7     122021|
+----------------------------------------+
```

3.

```
scala> val gp_1 = spark.sql("select gp_address.practice_code, gp_address.surgery_name, sum(gp_rx.items) AS nbr_prescriptions FROM gp_rx JOIN gp_address ON gp_rx.practice_code = gp_addr
ess.practice_code WHERE gp_address.postcode LIKE 'BL1%' OR gp_address.postcode LIKE 'BL2%' OR gp_address.postcode LIKE 'BL3%' GROUP BY gp_address.practice_code, gp_address.surgery_name
 ORDER BY gp_address.practice_code desc")
gp_1: org.apache.spark.sql.DataFrame = [practice_code: string, surgery_name: string ... 1 more field]

scala> gp_1.show()
+-------------+--------------------+----------------+
|practice_code|       surgery_name|nbr_prescriptions|
+-------------+--------------------+----------------+
|       Y04600|      BARDOC GP OOH|            3042|
|       Y03641|BOLTON COMMUNITY ...|            2267|
|       Y03366|OLIVE FAMILY PRAC...|            5030|
|       Y03364|GREAT LEVER PRACTICE|            5619|
|       Y03079|BOLTON COMMUNITY ...|           22209|
|       Y02943|NEUROLOGY LONG TE...|              56|
|       Y02790|BOLTON MEDICAL CE...|            4155|
|       Y02319|BOLTON GENERAL PR...|            5095|
|       Y00747|HALLIWELL HEALTH ...|             165|
|       Y00552|MINOR TREATMENT C...|               1|
|       Y00448|    DIABETES CENTRE|             150|
|       Y00233|        THE PARALLEL|               1|
|       Y00215|ORTHOPAEDIC & RHE...|              70|
|       Y00208|TIER 2 DERMATOLOG...|               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 SU...|            5443|
+-------------+--------------------+----------------+
only showing top 20 rows
```

```
scala> gp_1.write.format("JSON").mode("overwrite").option("header","true").save("hdfs:///user/2502010976/solution")

scala> val gp_2 = spark.read.format("JSON").option("header","true").load("hdfs:///user/2502010976/solution/")
gp_2: org.apache.spark.sql.DataFrame = [nbr_prescriptions: bigint, practice_code: string ... 1 more field]

scala> gp_2.show()
+-----------------+-------------+--------------------+
|nbr_prescriptions|practice_code|       surgery_name|
+-----------------+-------------+--------------------+
|             2267|       Y03641|BOLTON COMMUNITY ...|
|              165|       Y00747|HALLIWELL HEALTH ...|
|             9081|       P82018|THE ALASTAIR ROSS...|
|             9392|       P82607|WALMSLEY-CROMPTON...|
|               56|       Y02943|NEUROLOGY LONG TE...|
|             6191|       P82036|LITTLE LEVER HEAL...|
|             8710|       P82021|KIRBY-CROMPTON HE...|
|             6896|       P82020|LITTLE LEVER HEAL...|
|             5360|       P82633|GREAT LEVER HEALT...|
|             6108|       P82624|ORIENT HOUSE MEDI...|
|            10541|       P82613|SPRING VIEW MEDIC...|
|            22209|       Y03079|BOLTON COMMUNITY ...|
|            15775|       P82004|SWAN LANE MEDICAL...|
|               70|       Y00215|ORTHOPAEDIC & RHE...|
|             2618|       P82625|CHARLOTTE STREET ...|
|            12220|       P82023|MANDALAY MEDICAL ...|
|             9111|       P82011|TONGE FOLD HEALTH...|
|            14147|       P82009|ST HELENS ROAD PR...|
|            18419|       P82001|THE DUNSTAN PARTN...|
|             5095|       Y02319|BOLTON GENERAL PR...|
+-----------------+-------------+--------------------+
only showing top 20 rows
```

Essay

1. Suppose you encounter an OOM (Out Of Memory) issue while performing a broadcast join. What is likely cause this happen? How do you deal with this situation?
   = Out of Memory error pada saat menampilkan broadcast join disebabkan karena size estimator pada Spark yang terbatas. Ada beberapa cara untuk mengatasi hal tersebut, yang pertama, bisa dengan menggunakan 'ANALYZE TABLE (AWS | Azure)' untuk mengumpulkan detail dan menghitung statistik terkait dataframe sebelum bergabung (join). Yang kedua, cache tabel yang akan dibroadcast, jika broadacast join menampilkan BuildLeft, cache sisi kiri pada tabel, jika broadcast join menampilkan BuildRight, cache sisi kanan tabel. Yang ketiga, pada databricks runtime 7.0 ke atas, atur tipe join ke SortMergeJoin dengan mengaktifkan petunjuk join.

2. With Adaptive Query Execution, things become more comfortable for developers as Spark will do the partition coalescing automatically. What this statement means?
   = Adaptive Query Execution (AQE) adalah teknik pengoptimalan dalam Spark SQL yang memanfaatkan statistik runtime untuk memilih rencana eksekusi query yang peling efisien. Salah satu fitur utama di AQE adalah menyatukan partisi pasca shuffle. Fitur tersebut menyatukan partisi post-shuffle berdasarkan statistik map output. Fitur tersebut menyederhanakan penyetelan nomor partisi acak saat menjalankan query. Dalam hal ini, Spark dapat memilih nomor partisi acak yang tepat saat runtime setelah jumlah awal pastisi acak yang cukup besar ditetapkan.

3. Explain core parts of a structured streaming application as show in Figure 1. Suppose you are working with a file CSV Source

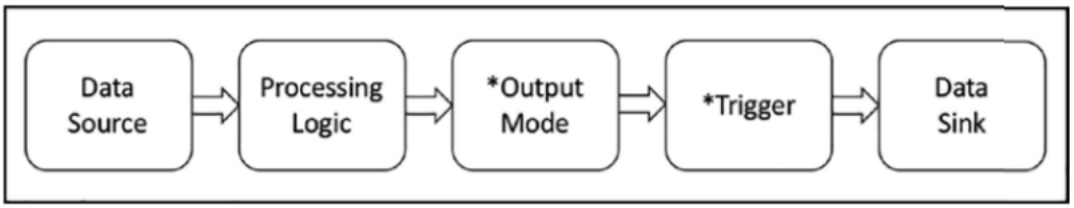

Figure 1

   = Pertama ada data source, pilihan data type yang disediakan oleh spark adalah file source, kafka source, socket source, dan rate source. Jika bekerja menggunakan sumber CSV berarti data source yang digunakan bertipe file source. Selanjutnya pada processing logic, hal yang dilakukan pertama adalah membuat schema, lalu buat dataframe dari streaming menggunakan metode readStream untuk membaca streaming data sebagai dataframe. Berikutnya terdapat output mode, output mode menentukan cara penulisan data pada sink. Terdapat 3 output mode yaitu append, complete, dan update. Selanjutnya terdapat trigger, trigger digunakan untuk menentukan kapan harus menjalankan logika komputasi streaming yang tersedia pada data streaming yang baru. Terdapat 4 tipe trigger yaitu, not specified (default), fixed interval, one-time, dan continous. Terakhir ada data sink yang digunakan untuk menyimpan output dari streaming application. Tipe sink yang didukung spark adalah file sink, kakfa sink, console sink, dan memory.