

Nama : Rafael Nicholas Tanaja

NIM : 2540118656

Link Video : https://binusianorg-my.sharepoint.com/personal/rafael_tanaja_binus_ac_id/_layouts/15/guestaccess.aspx?docid=0590541889f744da1a1ef23fae7595042&authkey=AdqJObDBe_oP5F759t5I66M&e=zTQtSt

Code Questions

Number 1

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

scala> import spark.implicits._
import spark.implicits._
```

```
scala> spark.sql("USE hr_db")
res7: org.apache.spark.sql.DataFrame = []

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

```
scala> val df = spark.sql("""
  | SELECT a.first_name, COUNT(*) as count, DENSE_RANK() OVER (ORDER BY COUNT(*) DESC) as rank
  | FROM hr_records a
  | WHERE a.gender LIKE "M"
  | GROUP BY a.first_name
  | """)
df: org.apache.spark.sql.DataFrame = [first_name: string, count: bigint ... 1 more field]
```

```
scala> df.show(15)
+-----+-----+-----+
|first_name|count|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
```

```
scala> val row = df.count()
row: Long = 1219
```

```
scala> df.write.format("csv").mode("overwrite").option("header", false).option("sep", "-").save("/user/2540118656/solution")
```

```
scala> verulam-blue ~ hdfs dfs -ls /user/2540118656/solution
Found 2 items
-rw-r--r--  1 verulam-blue supergroup      0 2023-01-31 15:12 /user/2540118656/solution/_SUCCESS
-rw-r--r--  1 verulam-blue supergroup 16712 2023-01-31 15:12 /user/2540118656/solution/part-00000-a25efc0b-5c0d-42b8-963b-632ca
b6585e8-c000.csv
```

```
scala> val test123 = spark.read.format("csv").option("header", "false").load("/user/2540118656/solution/")
test123: org.apache.spark.sql.DataFrame = [_c0: string]
```

```
scala> test123.show(15)
```

```
+-----+
|      _c0|
+-----+
| 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
```

Number 2

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

scala> emrdata.show(5)
+-----+-----+-----+-----+-----+-----+
|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|
+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

```
scala> emrdata.createOrReplaceTempView("emrdata1")
```

```
scala> val emr = spark.sql("""
  | SELECT MONTH(a.date_of_visit), SUM(a.ili_pne_admissions)
  | FROM emrdata1 a
  | WHERE MONTH(a.date_of_visit) = 5 OR MONTH(a.date_of_visit) = 6 OR MONTH(a.date_of_visit) = 7
  | GROUP BY MONTH(a.date_of_visit)
  | ORDER BY MONTH(a.date_of_visit) ASC
  | """)
emr: org.apache.spark.sql.DataFrame = [month(date_of_visit): int, sum(ili_pne_admissions): bigint]

scala> emr.show()
+-----+-----+
|month(date_of_visit)|sum(ili_pne_admissions)|
+-----+-----+
|          5|          200801|
|          6|          113289|
|          7|          122021|
+-----+-----+
```

```
scala> emr.write.format("csv").mode("overwrite").option("header", false).option("sep", "\t").save("/user/2540118656/solution")
```

```
scala> verulam-blue ~ hdfs dfs -ls /user/2540118656/solution
Found 4 items
-rw-r--r--  1 verulam-blue supergroup      0 2023-01-28 07:37 /user/2540118656/solution/_SUCCESS
-rw-r--r--  1 verulam-blue supergroup    9 2023-01-28 07:37 /user/2540118656/solution/part-00000-c63f0237-415c-4f12-aef0-3ae77
23f57ae-c000.csv
-rw-r--r--  1 verulam-blue supergroup    9 2023-01-28 07:37 /user/2540118656/solution/part-00001-c63f0237-415c-4f12-aef0-3ae77
23f57ae-c000.csv
-rw-r--r--  1 verulam-blue supergroup    9 2023-01-28 07:37 /user/2540118656/solution/part-00002-c63f0237-415c-4f12-aef0-3ae77
23f57ae-c000.csv
```

```
scala> val test123 = spark.read.format("csv").option("header", "false").load("/user/2540118656/solution/")
test123: org.apache.spark.sql.DataFrame = [_c0: string]
```

```
scala> test123.show()
```

```
+-----+
|   _c0|
+-----+
|5    200801|
|6    113289|
|7    122021|
+-----+
```

Number 3

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

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

```
scala> gprx.show(5)
```

sha	pct	practice_code	bnf_code	bnf_name	items	nic	act_cost	quantity	period
Q44	RJN	Y05218	0501013K0AAAJAJ	Co-Amoxiclav_Tab ...	1	3.59	3.33	21	201512
Q44	RJN	Y05218	0501130R0AAAAA	Nitrofurantoin_Ca...	1	14.39	13.42	28	201512
Q44	RTV	Y04937	0401020K0AAHAH	Diazepam_Tab 2mg	1	0.51	0.58	14	201512
Q44	RTV	Y04937	0401020P0AABAB	Lorazepam_Tab 1mg	1	2.65	2.46	28	201512
Q44	RTV	Y04937	0402010ABAAABAB	Quetiapine_Tab 25mg	2	2.01	2.08	84	201512

only showing top 5 rows

```
scala> gpaddress.show(5)
```

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 MEDIC...	QUEENS PARK MEDIC...	FARRER STREET	STOCKTON ON TEES	CLEVELAND	TS18 2AW
201512	A81003	VICTORIA MEDICAL ...	THE HEALTH CENTRE	VICTORIA ROAD	HARTLEPOOL	CLEVELAND	TS26 8DB
201512	A81004	WOODLANDS ROAD SU...	6 WOODLANDS ROAD		MIDDLESBROUGH	CLEVELAND	TS1 3BE
201512	A81005	SPRINGWOOD SURGERY	SPRINGWOOD SURGERY	RECTORY LANE	GUISBOROUGH		TS14 7DJ

only showing top 5 rows

```
scala> gprx.createOrReplaceTempView("gprx1")
```

```
scala> gpaddress.createOrReplaceTempView("gpaddress1")
```

```
scala> val gp = spark.sql("""
  | SELECT a.practice_code as practice_code, a.surgery_name as surgery_name, SUM(b.items) as nbr_prescriptions
  | FROM gpaddress1 a JOIN gprx1 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: org.apache.spark.sql.DataFrame = [practice_code: string, surgery_name: string ... 1 more field]
```

```
scala> gp.show(5)
```

```
+-----+-----+-----+
|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|
+-----+-----+-----+
```

only showing top 5 rows

```
scala> gp.write.mode("overwrite").json("/user/2540118656/solution")
```

```
scala> val test123 = spark.read.json("/user/2540118656/solution")
```

```
test123: org.apache.spark.sql.DataFrame = [nbr_prescriptions: bigint, practice_code: string ... 1 more field]
```

```
scala> test123.show(false)
```

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

Essay Question

1. Situasi ini dapat terjadi pada broadcast table yang terlalu besar untuk masuk ke dalam memori. Terdapat beberapa cara yang dapat dilakukan seperti mengurangi ukuran broadcast table, menggunakan metode join lainnya, menggunakan sampling techniques, atau menambahkan jumlah memori.
2. Adaptive Query Execution merupakan fitur di dalam Spark untuk query merubah eksekusi dari query secara dinamis berdasarkan distribusi data dan resources. Dengan AQE, query optimizer dapat mendeteksi secara otomatis dan mengatasi data distribusi yang skew. Fitur ini dapat membantu developer untuk melakukan partition coalescing secara otomatis. Fitur ini juga membuat query development menjadi lebih efisien dan mengurangi terjadinya error.

Reference : <https://www.databricks.com/blog/2020/05/29/adaptive-query-execution-speeding-up-spark-sql-at-runtime.html>

3. Data Source: Bagian ini merupakan tempat untuk mengambil data. Jika menggunakan CSV maka akan diambil melalui File Source.

Processing Logic: Di dalam bagian ini merupakan bagian dimana user akan mendefinisikan operasi yang akan dilakukan kepada data.

Output Mode: Pada Output Mode, akan mengeluarkan hasil query yang akan ditulis ke dalam sink.

Trigger: Pada Trigger, bagian ini merupakan tempat mengontrol kapan akan mengeksekusi processing logic pada streaming, seperti akan menghentikan operasi jika jumlah data sudah terpenuhi atau saat waktu yang ditentukan telah selesai.

Data Sink: Bagian ini berguna untuk menyimpan hasil output dari streaming application. Pada kasus ini akan disimpan pada file sink dalam bentuk CSV.