CSCI 5408 – Data Management, Warehousing and Analytics – Assignment 3

TASK DESCRIPTION

- ✓ To understand Apache Spark concepts and RDD
- ✓ To Write spark application to preprocess and Count the distinct words in a given document.
- ✓ To Write a spark application to query the Baby names database and save results.
- ✓ To Write a spark application to query the NYPD Motor vehicles collision database.

TASK 1 - SPARK DESIGN - INSTALLATION STEPS

- ✓ Dependency check – Java is already installed in the machine
- ✓ Spark is downloaded and installed in "C:\Spark" folder
- ✓ Downloading winutils and storing in C drive.
- ✓ Scala is downloaded and installed
- ✓ Environmental variables are set – SPARK_HOME, HADOOP_HOME.

User variables for Yamuna

Variable	Value
HADOOP_HOME	C:\winutils
JAVA_HOME	C:\Program Files\Java\jdk1.8.0_111
Path	C:\Users\Yamuna\Anaconda2;C:\Users\Yamuna\Anaconda2\Scrip...
SPARK_HOME	C:\Spark
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

Verifying Spark Installation

```
C:\Spark\bin>spark-shell
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
17/02/21 17:37:16 WARN SparkContext: Use an existing SparkContext, some configuration may not take effect.
Spark context Web UI available at http://169.254.83.3:4040
Spark context available as 'sc' (master = local[*], app id = local-1487713036524).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.0.2
      /_/

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_111)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

TASK 2 – TO COUNT DISTINCT WORDS USING APACHE SPARK

- ✓ The given dataset is pre-processed to remove all special characters like (‘,’,’.’,’;’,’:’,’-‘)
- ✓ Regular expression is used for pre-processing.
- ✓ Text is converted to lower case before performing the count.

```
(u'hordes', 1)
(u'child', 1)
(u'predestind', 1)
(u'yellow', 2)
(u'four', 2)
(u'hath', 29)
(u'sleep', 1)
(u'perverted', 1)
(u'appetite', 1)
(u'reechod', 1)
(u'whose', 22)
(u'believst', 1)
(u'trangressd', 1)
(u'under', 4)
(u'lore', 1)
(u'lord', 1)
(u'sped', 2)
(u'pride', 1)
(u'sway', 1)
(u'worth', 5)
(u'sinking', 1)
(u'amorous', 1)
(u'rescue', 1)
(u'void', 5)
```

TASK 3 – TO QUERY BABY NAMES DATABASE AND FETCH THE OUTPUT

Query 1 - Total number of birth registered in a year

BirthbyYear=sqlc.sql("Select Year, sum(Count) as NumberOfBirth from Births where Year = 1991").show()

```
17/02/21 16:39:28 INFO DAGScheduler: Job 3 finished: showString at NativeMethodAccessorImpl.java:-2, took 7.693875 s
17/02/21 16:39:28 INFO CodeGenerator: Code generated in 7.685914 ms
+----+-------------+
|Year|NumberOfBirth|
+----+-------------+
|1903|     381207.0|
|1953|    3850103.0|
|1897|     346960.0|
|1957|    4200026.0|
|1880|     201484.0|
|1987|    3603553.0|
|1956|    4121206.0|
|1936|    2077176.0|
|2012|    3643336.0|
|1958|    4131596.0|
|1910|     590719.0|
|1943|    2821928.0|
|1915|    1832477.0|
|1972|    3143851.0|
|1931|    2103624.0|
|1911|     644267.0|
|1926|    2295809.0|
|1938|    2212118.0|
|1988|    3692441.0|
|1918|    2171184.0|
+----+-------------+
only showing top 20 rows
```

Query 2 - Total number of birth registered in a year by gender

BirthbyGender=sqlc.sql("Select Year, Gender, sum(Count) from Births group by Gender, Year ").show()

```
17/02/21 16:44:55 INFO DAGScheduler: Job 3 finished: showString at NativeMethodAccessorImpl.java:-2, took 0.784314 s
17/02/21 16:44:55 INFO CodeGenerator: Code generated in 15.444519 ms
+----+------+------------+
|Year|Gender|NumberOfBirth|
+----+------+------------+
|1966|     M|    1783964.0|
|1978|     M|    1642250.0|
|1919|     F|    1130145.0|
|1928|     F|    1153117.0|
|1913|     F|     624518.0|
|2007|     M|    2072139.0|
|1926|     F|    1185304.0|
|1921|     M|    1101457.0|
|1887|     F|     145982.0|
|1939|     M|    1106544.0|
|1904|     F|     275371.0|
|1908|     F|     334313.0|
|2014|     M|    1901376.0|
|1954|     F|    1941682.0|
|1935|     F|    1048428.0|
|1981|     F|    1667465.0|
|1996|     F|    1752249.0|
|1900|     F|     299828.0|
|1925|     F|    1217352.0|
|1996|     M|    1893378.0|
+----+------+------------+
only showing top 20 rows
```

Query 3 - Input a year and populate top 5 most popular names registered that year

Mostpopularname=sqlc.sql("select Count as NumberOfPeopleWithName, Name from births where Year = "+InYear+" order by count desc limit 5").show()

Note: Year passed as input while running the code

```
17/02/21 16:14:58 INFO CodeGenerator: Code generated in 8.194358 ms
17/02/21 16:14:59 INFO CodeGenerator: Code generated in 7.449272 ms
+----------------------+--------+
|NumberOfPeopleWithName|    Name|
+----------------------+--------+
|                   996|Kristine|
|                   994|   Tracy|
|                    99|   Deena|
|                    99|Estefani|
|                    99|  Kiesha|
+----------------------+--------+

17/02/21 16:14:59 INFO SparkSqlParser: Parsing command: Collisions
17/02/21 16:14:59 INFO SparkContext: Invoking stop() from shutdown hook
17/02/21 16:14:59 INFO SparkUI: Stopped Spark web UI at http://169.254.83.3:4040
```

Query 4 - Input a name and populate total number of birth registration throughout the dataset for that name

TotalbirthRegistration = sqlc.sql("select sum(Count) as TotalBirthRegistration from births where Name = '"+InName+"'").show()

Note: Name passed as input while running the code

```
17/02/21 16:36:17 INFO TaskSchedulerImpl: Removed TaskSet 4.0, whose tasks have all completed, from pool
17/02/21 16:36:17 INFO CodeGenerator: Code generated in 15.054593 ms
+---------------------+
|TotalBirthRegistration|
+---------------------+
|            4130441.0|
+---------------------+
```

TASK 4 – TO QUERY NYPD MOTOR VEHICLES COLLISION DATABASE AND FETCH THE OUTPUT

Schema

```
CollisionSchema = StructType([\
StructField("INCIDENTDATE", StringType(), True),\
StructField("INCIDENTTIME", TimestampType(), True),\
StructField("BOROUGH", StringType(), True),\
StructField("ZIPCODE", StringType(), True),\
StructField("LATITUDE", StringType(), True),\
StructField("LONGITUDE", StringType(), True),\
StructField("LOCATION", StringType(), True),\
StructField("ONSTREETNAME", StringType(), True),\
StructField("CROSSSTREETNAME", StringType(), True),\
StructField("OFFSTREETNAME", StringType(), True),\
StructField("NUMBEROFPERSONSINJURED", IntegerType(), True),\
StructField("NUMBEROFPERSONSKILLED", IntegerType(), True),\
StructField("NUMBEROFPEDESTRIANSINJURED", IntegerType(), True),\
StructField("NUMBEROFPEDESTRIANSKILLED", IntegerType(), True),\
StructField("NUMBEROFCYCLISTINJURED", IntegerType(), True),\
StructField("NUMBEROFCYCLISTKILLED", IntegerType(), True),\
StructField("NUMBEROFMOTORISTINJURED", IntegerType(), True),\
StructField("NUMBEROFMOTORISTKILLED", IntegerType(), True),\
StructField("CONTRIBUTINGFACTORVEHICLE1", StringType(), True),\
StructField("CONTRIBUTINGFACTORVEHICLE2", StringType(), True),\
StructField("CONTRIBUTINGFACTORVEHICLE3", StringType(), True),\
StructField("CONTRIBUTINGFACTORVEHICLE4", StringType(), True),\
StructField("CONTRIBUTINGFACTORVEHICLE5", StringType(), True),\
StructField("UNIQUEKEY", StringType(), True),\
StructField("VEHICLETYPECODE1", StringType(), True),\
StructField("VEHICLETYPECODE2", StringType(), True),\
StructField("VEHICLETYPECODE3", StringType(), True),\
StructField("VEHICLETYPECODE4", StringType(), True),\
StructField("VEHICLETYPECODE5", StringType(), True)])
```

Query 1 - Capture total injuries and fatalities associated with each motor collision record(identified by a unique incident key)

```
TotalMotorInjuries = sqlc.sql("select UNIQUEKEY, (sum(NUMBEROFMOTORISTINJURED) +
sum(NUMBEROFMOTORISTKILLED))  as TotalMotoristInjuries\

 from Collisions group by UNIQUEKEY").show()
```

```
17/02/21 16:45:01 INFO DAGScheduler: Job 4 finished: showString at NativeMethodAccessorImpl.java:-2, took 5.658005 s
17/02/21 16:45:01 INFO CodeGenerator: Code generated in 19.130439 ms
+---------+--------------------+
|UNIQUEKEY|TotalMotoristInjuries|
+---------+--------------------+
|  3284306|                   0|
|  3442948|                   0|
|  3441699|                   0|
|  3442340|                   0|
|  3528339|                   0|
|  3437474|                   0|
|  3511855|                   0|
|  3437121|                   1|
|  3508234|                   0|
|  3545205|                   0|
|  3525673|                   2|
|  3433820|                   1|
|  3433033|                   0|
|  3600440|                   0|
|  3433296|                   0|
|  3600968|                   0|
|  3599752|                   0|
|  3598415|                   0|
|  3431467|                   0|
|  3429576|                   0|
+---------+--------------------+
only showing top 20 rows
```

Query 2 - Capture total incident counts in a year (grouped by year)

TotalIncidentsYear = sqlc.sql("SELECT Year(DATE_FORMAT(CAST(UNIX_TIMESTAMP(INCIDENTDATE, 'mm/dd/yyyy') AS TIMESTAMP), 'yyyy-mm-dd')) as IncidentYear,

(count(NUMBEROFMOTORISTINJURED) + count(NUMBEROFMOTORISTKILLED)

+ count(NUMBEROFPERSONSINJURED) + count(NUMBEROFPERSONSKILLED) + count(NUMBEROFPEDESTRIANSINJURED)

*+ count(NUMBEROFCYCLISTINJURED) + count(NUMBEROFCYCLISTKILLED)) as TotalIncidentCounts *

*from Collisions *

group by Year(DATE_FORMAT(CAST(UNIX_TIMESTAMP(INCIDENTDATE, 'mm/dd/yyyy') AS TIMESTAMP), 'yyyy-mm-dd'))").show()

```
17/02/21 15:33:27 INFO DAGScheduler: Job 1 finished: showString at NativeMethodAccessorImpl.java:-2, took 9.110608 s
17/02/21 15:33:27 INFO CodeGenerator: Code generated in 6.40078 ms
+------------+-------------------+
|IncidentYear|TotalIncidentCounts|
+------------+-------------------+
|        2015|            1522773|
|        2013|            1425823|
|        2014|            1441503|
|        2012|             703654|
|        2016|            1590841|
|        2017|             110320|
+------------+-------------------+
```

Query 3 - Capture total injuries (can be sum of injuries and fatalities) grouped by year and quarter

TotalInjuriesQuarter = sqlc.sql("SELECT Year(DATE_FORMAT(CAST(UNIX_TIMESTAMP(INCIDENTDATE, 'mm/dd/yyyy') AS TIMESTAMP), 'yyyy-mm-dd')) as IncidentYear,

*Quarter(DATE_FORMAT(CAST(UNIX_TIMESTAMP(INCIDENTDATE, 'mm/dd/yyyy') AS TIMESTAMP), 'yyyy-mm-dd')) as IncidentQuarter, *

(sum(NUMBEROFMOTORISTINJURED) + sum(NUMBEROFMOTORISTKILLED)

+ sum(NUMBEROFPERSONSINJURED) + sum(NUMBEROFPERSONSKILLED) + sum(NUMBEROFPEDESTRIANSINJURED)

*+ sum(NUMBEROFCYCLISTINJURED) + sum(NUMBEROFCYCLISTKILLED)) as SumOfInjuries *

*from Collisions *

group by Year(DATE_FORMAT(CAST(UNIX_TIMESTAMP(INCIDENTDATE, 'mm/dd/yyyy') AS TIMESTAMP), 'yyyy-mm-dd')),

Quarter(DATE_FORMAT(CAST(UNIX_TIMESTAMP(INCIDENTDATE, 'mm/dd/yyyy') AS TIMESTAMP), 'yyyy-mm-dd'))").show()

```
17/02/21 15:54:43 INFO TaskSchedulerImpl: Removed TaskSet 3.0, whose tasks have all completed, from pool
17/02/21 15:54:43 INFO DAGScheduler: Job 1 finished: showString at NativeMethodAccessorImpl.java:-2, took 7.301913 s
17/02/21 15:54:43 INFO CodeGenerator: Code generated in 7.567 ms
+------------+--------------+-------------+
|IncidentYear|IncidentQuarter|SumOfInjuries|
+------------+--------------+-------------+
|        2015|             2|        26906|
|        2014|             4|        26257|
|        2013|             2|        29223|
|        2012|             4|        26392|
|        2016|             2|        31305|
|        2014|             1|        21834|
|        2013|             3|        29968|
|        2015|             4|        27549|
|        2013|             4|        27774|
|        2014|             3|        27206|
|        2014|             2|        27495|
|        2016|             1|        23716|
|        2016|             4|        32324|
|        2015|             3|        28178|
|        2012|             3|        28700|
|        2013|             1|        23677|
|        2015|             1|        20396|
|        2017|             1|         8602|
|        2016|             3|        41695|
```

Query 4 - Capture total injuries (sum of injuries and fatalities) and incident count grouped by Borough, year and month

TotalInjuriesMYB = sqlc.sql("SELECT BOROUGH, Year(DATE_FORMAT(CAST(UNIX_TIMESTAMP(INCIDENTDATE, 'mm/dd/yyyy') AS TIMESTAMP), 'yyyy-mm-dd')) as IncidentYear,

*month(DATE_FORMAT(CAST(UNIX_TIMESTAMP(INCIDENTDATE, 'mm/dd/yyyy') AS TIMESTAMP), 'yyyy-mm-dd')) as IncidentMonth, *

(sum(NUMBEROFMOTORISTINJURED) + sum(NUMBEROFMOTORISTKILLED)

+ sum(NUMBEROFPERSONSINJURED) + sum(NUMBEROFPERSONSKILLED) + sum(NUMBEROFPEDESTRIANSINJURED)

*+ sum(NUMBEROFCYCLISTINJURED) + sum(NUMBEROFCYCLISTKILLED)) as SumOfInjuries *

*from Collisions *

group by Year(DATE_FORMAT(CAST(UNIX_TIMESTAMP(INCIDENTDATE, 'mm/dd/yyyy') AS TIMESTAMP), 'yyyy-mm-dd')),

month(DATE_FORMAT(CAST(UNIX_TIMESTAMP(INCIDENTDATE, 'mm/dd/yyyy') AS TIMESTAMP), 'yyyy-mm-dd')), BOROUGH ").show()

```
17/02/21 15:57:38 INFO DAGScheduler: Job 1 finished: showString at NativeMethodAccessorImpl.java:-2, took 7.237182 s
17/02/21 15:57:38 INFO CodeGenerator: Code generated in 6.038904 ms
+-------------+------------+-------------+-------------+
|      BOROUGH|IncidentYear|IncidentMonth|SumOfInjuries|
+-------------+------------+-------------+-------------+
|     BROOKLYN|        2016|            2|         1888|
|        BRONX|        2015|           11|          973|
|       QUEENS|        2015|            2|         1016|
|         null|        2012|           10|         2118|
|STATEN ISLAND|        2016|           12|          237|
|       QUEENS|        2014|            5|         1803|
|     BROOKLYN|        2014|            3|         2223|
|    MANHATTAN|        2016|            2|          919|
|STATEN ISLAND|        2017|            1|          293|
|         null|        2015|            2|         2011|
|       QUEENS|        2016|            3|         1952|
|        BRONX|        2013|           11|         1075|
|     BROOKLYN|        2015|            8|         2585|
|    MANHATTAN|        2016|            7|         1061|
|STATEN ISLAND|        2013|            5|          420|
|    MANHATTAN|        2016|           11|         1014|
|    MANHATTAN|        2013|            1|         1115|
|    MANHATTAN|        2015|            1|          824|
|STATEN ISLAND|        2014|            7|          260|
|        BRONX|        2012|           12|         1046|
+-------------+------------+-------------+-------------+
only showing top 20 rows
```

SUMMARY

All the given tasks are performed and the output files are stored. The Stored output files are attached with this report. The processing of NYPD Motor vehicles collision database was a bit challenging as the Field names were having space and those were manually removed before processing.

REFERENCE

[1] Paul Hernandez, "Apache Spark Installation", 2016,

https://hernandezpaul.wordpress.com/2016/01/24/apache-spark-installation-on-windows-10/

[2] Stack Overflow Community, "String to Date Conversion", 2015,
http://stackoverflow.com/questions/40763796/convert-date-from-string-to-date-format-in-dataframes

[3] Stack Overflow Community, "Removing Special Characters", 2015,

http://stackoverflow.com/questions/5843518/remove-all-special-characters-punctuation-and-spaces-from-string