Προχωρημένα Θέματα Βάσεων Δεδομένων

Ονοματεπώνυμο:	Μπάρμπα Παναγιώτα-Νικολέττα
ΑΜ :	03118604
Εξάμηνο :	11ο
Ομάδα :	41
Github :	Github Link

Ζητούμενο 1

Η εγκατάσταση και διαμόρφωση της πλατφόρμας εκτέλεσης Apache Spark ώστε να εκτελείται πάνω από το διαχειριστή πόρων του Apache Hadoop, YARN, έγινε σε 2 εικονικά μηχανήματα σε τοπικό μηχάνημα (δεν χρησιμοποιήθηκε το cloud service okeanos). Η διαμόρφωση των εργαλείων που χρησιμοποιήθηκαν περιγράφεται στο README αρχείο του Github αποθετηρίου.

Οι web διεπαφές των Apache Spark και Apache Hadoop είναι προσβάσιμες από τους παρακάτω συνδέσμους:

- Apache Spark : http://192.168.64.9:8080/
- Apache Hadoop : http://192.168.64.9:9870/
- Apache Hadoop YARN : http://192.168.64.9:8088/

Ζητούμενο 2

Αυτό όπως και τα επόμενα ζητούμενα υλοποιήθηκαν με χρήση της γλώσσας προγραμματισμού Python3 και του PySpark.

Δημιουργήθηκε ένα DataFrame από το βασικό σύνολο δεδομένων και διατηρώντας τα ονόματα των στηλών, προσαρμόστηκαν οι τύποι ορισμένων στηλών ως εξής:

- Date Rptd : string → date
- DATE OCC : string → date
- Vict Age : string → integer
- LAT : double → double
- LON : double → double

Επίσης, στο αρχειο IncomeData2015.csv η στήλη "Estimated Median Income" έχει τύπο string της μορφής: '$number' , οπότε αφαιρέθηκε το '$' και έγινε μετατροπή σε integer.

Τέλος, ενώθηκαν τα DataFrame που περιέχουν τα δεδομένα καταγραφής εγκλημάτων για το Los Angeles από το 2010 μέχρι το 2019 και από το 2020 μέχρι σήμερα, τα δεδομένα με reverse geocoding πληροφορία και τα δεδομένα σχετικά με το μέσο εισόδημα ανά νοικοκυριό και ταχυδρομικό κώδικα δημιουργώντας ένα νέο DataFrame, το οποίο στην συνέχεια αποθηκέυτηκε.

Ο συνολικός αριθμός γραμμών και ο τύπος κάθε στήλης φαίνονται παρακάτω:

```
Total Rows: 3001575
root
 |-- DR_NO: integer (nullable = true)
 |-- Date Rptd: date (nullable = true)
 |-- DATE OCC: date (nullable = true)
 |-- TIME OCC: integer (nullable = true)
 |-- AREA : integer (nullable = true)
 |-- AREA NAME: string (nullable = true)
 |-- Rpt Dist No: integer (nullable = true)
 |-- Part 1-2: integer (nullable = true)
 |-- Crm Cd: integer (nullable = true)
 |-- Crm Cd Desc: string (nullable = true)
 |-- Mocodes: string (nullable = true)
 |-- Vict Age: integer (nullable = true)
 |-- Vict Sex: string (nullable = true)
 |-- Vict Descent: string (nullable = true)
 |-- Premis Cd: integer (nullable = true)
 |-- Premis Desc: string (nullable = true)
 |-- Weapon Used Cd: integer (nullable = true)
 |-- Weapon Desc: string (nullable = true)
 |-- Status: string (nullable = true)
 |-- Status Desc: string (nullable = true)
 |-- Crm Cd 1: integer (nullable = true)
 |-- Crm Cd 2: integer (nullable = true)
 |-- Crm Cd 3: integer (nullable = true)
 |-- Crm Cd 4: integer (nullable = true)
 |-- LOCATION: string (nullable = true)
 |-- Cross Street: string (nullable = true)
 |-- LAT: double (nullable = true)
 |-- LON: double (nullable = true)
 |-- AREA: integer (nullable = true)
 |-- ZIPcode: string (nullable = true)
 |-- Community: string (nullable = true)
 |-- Estimated Median Income: integer (nullable = true)
```

Ζητούμενο 3

Το Query 1 υλοποιήθηκε χρησιμοποιώντας τα DataFrame και SQL API με 4 Spark Executors. Οι δύο υλοποιήσεις βρίσκονται σε δύο διαφορετικά αρχεία και προφανώς δίνουν το ίδιο αποτέλεσμα αλλα σε διαφορετικούς χρόνους εκτέλεσης[1]. Γεγονός ελαφρώς αναμενόμενο αφού η βιβλιογραφία αναφέρει ότι το DataFrame API έχει καλύτερη επίδοση για πιο περίπλοκα ερωτήματα. Σε κάθε περίπτωση, όμως, οι χρόνοι εκτέλεσης δεν διαφέρουν πολύ, καθώς και τα δύο APIs χρησιμοποιούν το ίδιο execution plan και το ίδιο query optimizer. Συνεπώς, παίζει ρόλο και η εξοικείωση του προγραμματιστή με το κάθε API, σημείο που υπερτερεί το DataFrame API διότι προσφέρει μεγαλύτερη ευκολία και έλεγχο. Οι χρόνοι εκτέλεσης και τα αποτελέσματα φαίνονται παρακάτω:

```
Query 1 Dataframe Execution Time:  0.0834348201751709

Query 1 SQL Execution Time: 0.11646175384521484

===== Query 1 SQL Result =====
+----+-----+----------+----+
|Year|Month|CrimeCount|Rank|
+----+-----+----------+----+
|2010|1    |19517     |1   |
|2010|3    |18131     |2   |
|2010|7    |17856     |3   |
|2011|1    |18138     |1   |
|2011|7    |17283     |2   |
|2011|10   |17034     |3   |
|2012|1    |17946     |1   |
|2012|8    |17661     |2   |
|2012|5    |17502     |3   |
|2013|8    |17441     |1   |
|2013|1    |16822     |2   |
|2013|7    |16644     |3   |
|2014|10   |17329     |1   |
|2014|7    |17258     |2   |
|2014|12   |17198     |3   |
|2015|10   |19220     |1   |
|2015|8    |19011     |2   |
|2015|7    |18709     |3   |
|2016|10   |19659     |1   |
|2016|8    |19491     |2   |
|2016|7    |19448     |3   |
|2017|10   |20433     |1   |
|2017|7    |20193     |2   |
|2017|1    |19835     |3   |
|2018|5    |19974     |1   |
|2018|7    |19876     |2   |
```

[1]Να σημειωθεί ότι οι χρόνοι εκτέλεσης αφορούν αποκλειστικά την διάρκεια εκτέλεσης των ερωτημάτων και δεν προσμετράται το φόρτωμα και η εκτύπωση των στοιχείων

```
|2018|8    |19762     |3   |
|2019|7    |19123     |1   |
|2019|8    |18980     |2   |
|2019|3    |18859     |3   |
|2020|1    |18510     |1   |
|2020|2    |17257     |2   |
|2020|5    |17211     |3   |
|2021|10   |19311     |1   |
|2021|7    |18663     |2   |
|2021|8    |18379     |3   |
|2022|5    |20428     |1   |
|2022|10   |20285     |2   |
|2022|6    |20221     |3   |
|2023|8    |19842     |1   |
|2023|1    |19789     |2   |
|2023|7    |19777     |3   |
|2024|1    |6709      |1   |
+----+-----+----------+----+
```

Ζητούμενο 4

Αντίστοιχα με παραπάνω, το Query 2 υλοποιήθηκε χρησιμοποιώντας τα DataFrame, SQL και RDD API με 4 Spark Executors. Οι τρεις υλοποιήσεις βρίσκονται σε τρία διαφορετικά αρχεία. Είναι προφανές ότι ο χρόνος εκτέλεσης του RDD API είναι πολύ μεγαλύτερος από τους άλλους δύο. Αυτό οφείλεται στο γεγονός ότι το RDD API είναι πιο χαμηλού επιπέδου και απαιτεί περισσότερη ¨χειρωνακτική' επεξεργασία από τον προγραμματιστή. Αντίθετα, το DataFrame API και το SQL API προσφέρουν υψηλότερου επιπέδου εργαλεία και εκτελούν μια σειρά από βελτιστοποιήσεις στον κώδικα προτού εκτελεστούν, δίνοντας έτσι την βέλτιστη δομή σε αυτόν. Ακόμα, από την στιγμή που επιθυμείται η παραλλήλοποιηση της εκτέλεσης του ερωτήματος(4 Spark Executors), η καταλληλότερη δομή είναι το DataFrame API αφού είναι πολύ αποδοτικό στην διαχείριση πόρων.Οι χρόνοι εκτέλεσης και τα αποτελέσματα φαίνονται παρακάτω:

```
Query 2 Dataframe Execution Time:  0.06137490272521973
Query 2 SQL Execution Time:  0.09846782684326172

Query 2 RDD Execution Time: 15.420125961303711

===== Query 2 RDD Result =====
+-----------+----------+
|Time of Day|CrimeCount|
+-----------+----------+
|Night      |237692    |
|Evening    |187172    |
|Afternoon  |147657    |
|Morning    |123371    |
+-----------+----------+
```

Ζητούμενο 5

Το Query 3 υλοποιήθηκε χρησιμοποιώντας τα DataFrame και SQL API με 2,3 και Spark Executors. Όπως φαίνεται από τα αποτελέσματα όσο αυξάνει ο αριθμός των Spark Executors τόσο μειώνεται ο χρόνος εκτέλεσης. Βέβαια, είναι σημαντικό να ληφθεί υπόψη ότι η αύξηση των Spark Executors δεν οδηγεί απαραίτητα σε γραμμική μείωση του χρόνου εκτέλεσης. Αυτό οφείλεται στο γεγονός ότι η αύξηση των Spark Executors αυξάνει τον αριθμό των tasks που εκτελούνται παράλληλα, αλλά αυξάνει επίσης τον αριθμό των data shuffles που πρέπει να εκτελεστούν, πράγμα ανεπιθύμιτο. Συνεπώς, η βέλτιστη επιλογή του αριθμού των Spark Executors εξαρτάται από την φύση των δεδομένων και την φύση του ερωτήματος. Οι χρόνοι εκτέλεσης και τα αποτελέσματα φαίνονται παρακάτω:
Query 3 Dataframe Execution Time: 5.956484794616699with 2 executors

Query 3 Dataframe Execution Time: 3.3060457706451416with 3 executors

Query 3 Dataframe Execution Time: 2.7904648780822754with 4 executors

```
Query 3 SQL Execution Time: 5.258688926696777with 2 executors

Query 3 SQL Execution Time: 2.540524959564209with 3 executors

Query 3 SQL Execution Time: 2.130349636077881with 4 executors

===== Query 3 SQL Result =====
+--------------------+-----+
|        Vict Descent|Count|
+--------------------+-----+
|Hispanic/Latin/Me...| 1053|
|               White|  610|
|               Black|  349|
|               Other|  272|
|             Unknown|   71|
|         Other Asian|   46|
|              Korean|    4|
|             Chinese|    1|
|American Indian/A...|    1|
+--------------------+-----+
```