Διαχείριση Δεδομένων Μεγάλης Κλίμακας - Αναφορά Εργασίας

Διολέτης Μηνάς Προμπονάς Αντώνιος

Ιούνιος 2024

1 Διευκρινίσεις

Μπορείτε να βρείτε τα πηγαία αρχεία για των υλοποιήσεων για όλα τα σχετικά ζητήματα στον ακόλουθο σύνδεσμο στο github:

`https://github.com/minasd1/big_data_management`
Παρακάτω παρουσιάζονται οι κατάλληλες επεξηγήσεις για όσα ζητούμενα απαιτώνται.

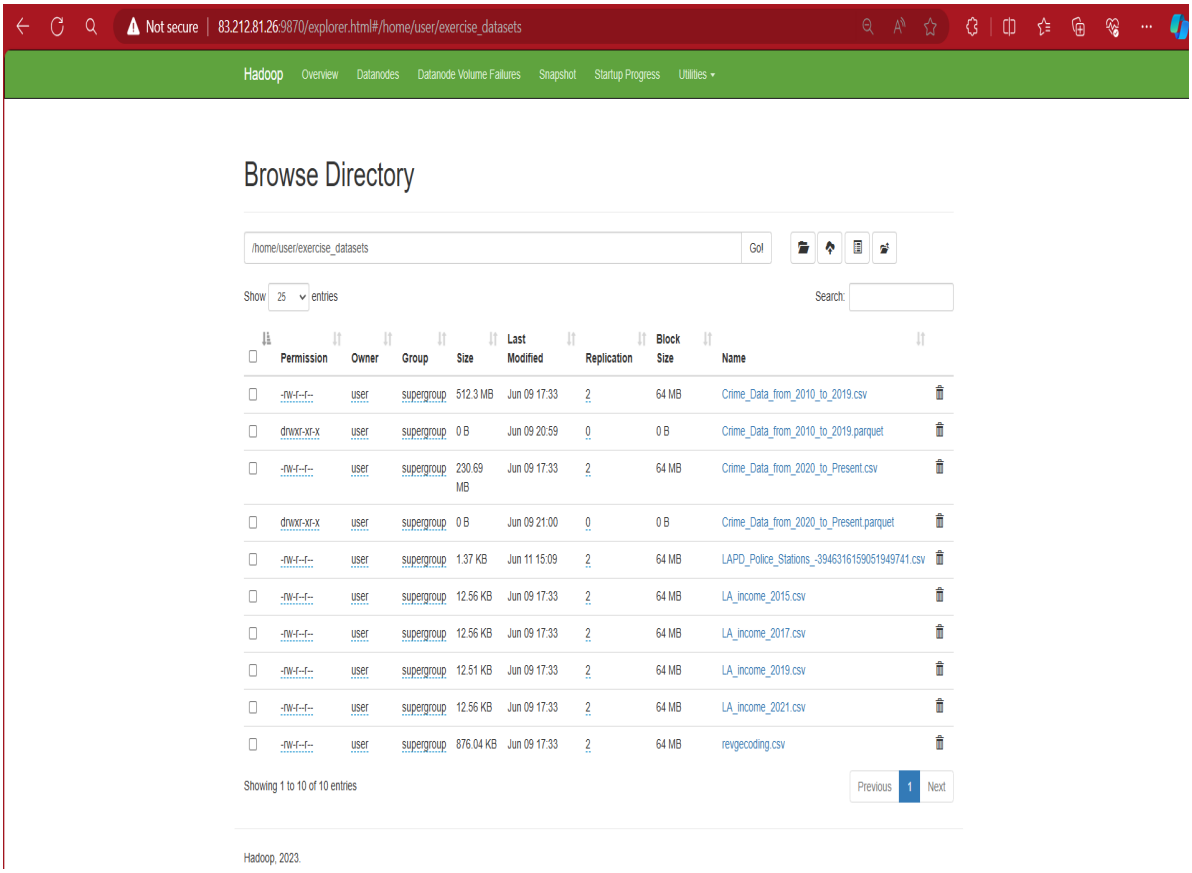
2 Ζητούμενο 2

Αρχικά, φορτώθηκαν τοπικά τα δοθέντα αρχεία. Έπειτα, με χρήση του WinScp μεταφέρθηκαν στον κόμβο master. Στη συνέχεια, εκτελέστηκαν διαδοχικά οι ακόλουθες εντολές.

```
hadoop fs -mkdir -p ~/exercise_datasets/
hadoop fs -put *.csv ~/exercise_datasets
hadoop fs -ls ~/exercise_datasets/
```

Βάσει των παραπάνω, δημιουργήθηκε ένας φάκελος hdfs, τοποθετήθηκαν μέσα σε αυτόν όλα τα αρχεία ενδιαφέροντος, και εξετάστηκε η ορθή τους τοποθέτηση.

Η κατάσταση του συστήματος αχείων με τα δεδομένα διαθέσιμα φαίνεται στην εικόνα 1.



Σχήμα 1: Στιγμιότυπο οθόνης με την κατάσταση του συστήματος αρχείων

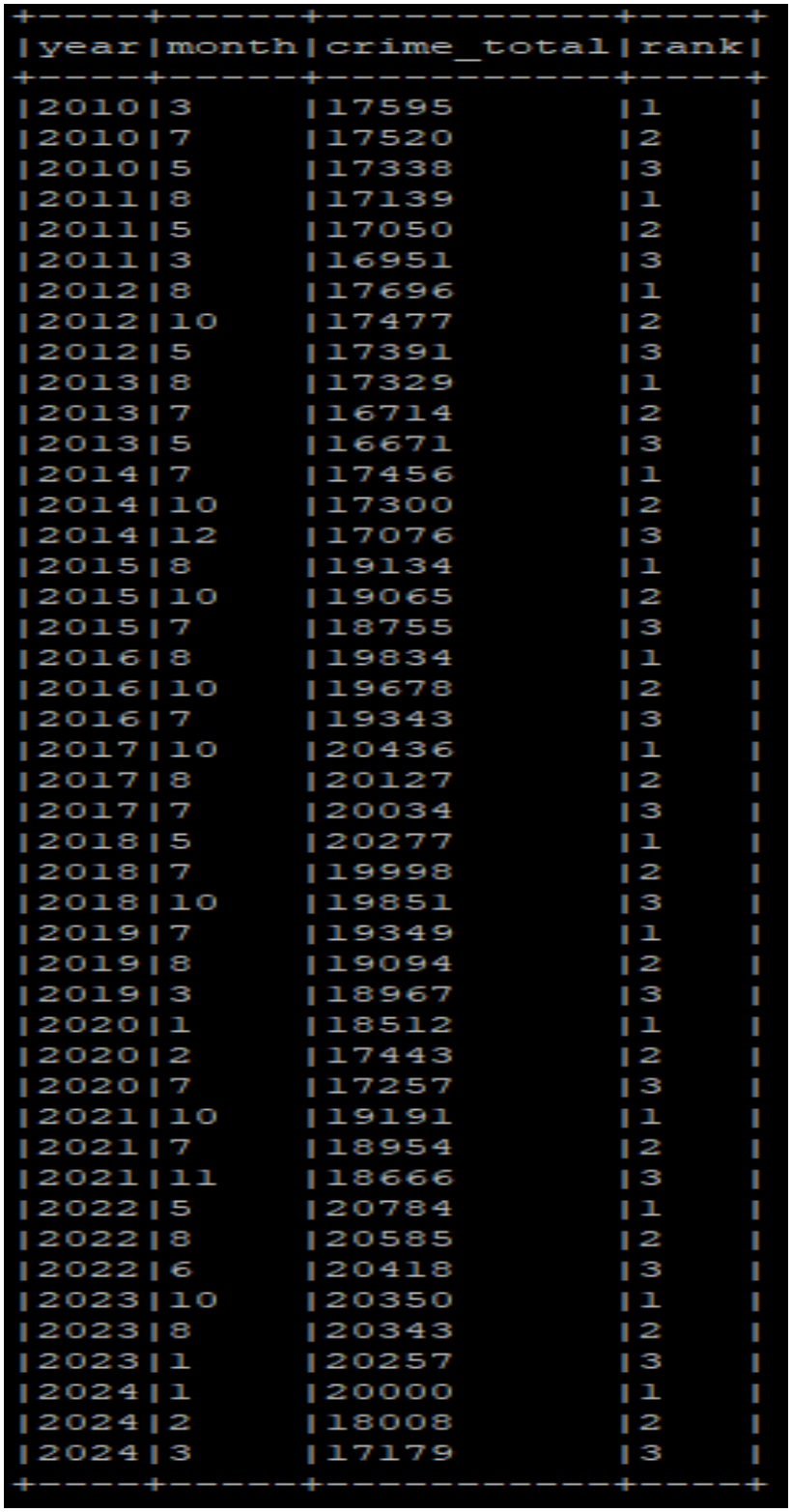
Ο κώδικας μετατροπής των ζητούμενων αρχείων από csv σε parquet είναι επίσης αναρτημένος στο σύνδεσμο του github.

3 Ζητούμενο 3

Παρακάτω παρουσιάζονται τα αποτελέσματα για τις διαφορετικές εκτελέσεις του ζητουμένου 3.

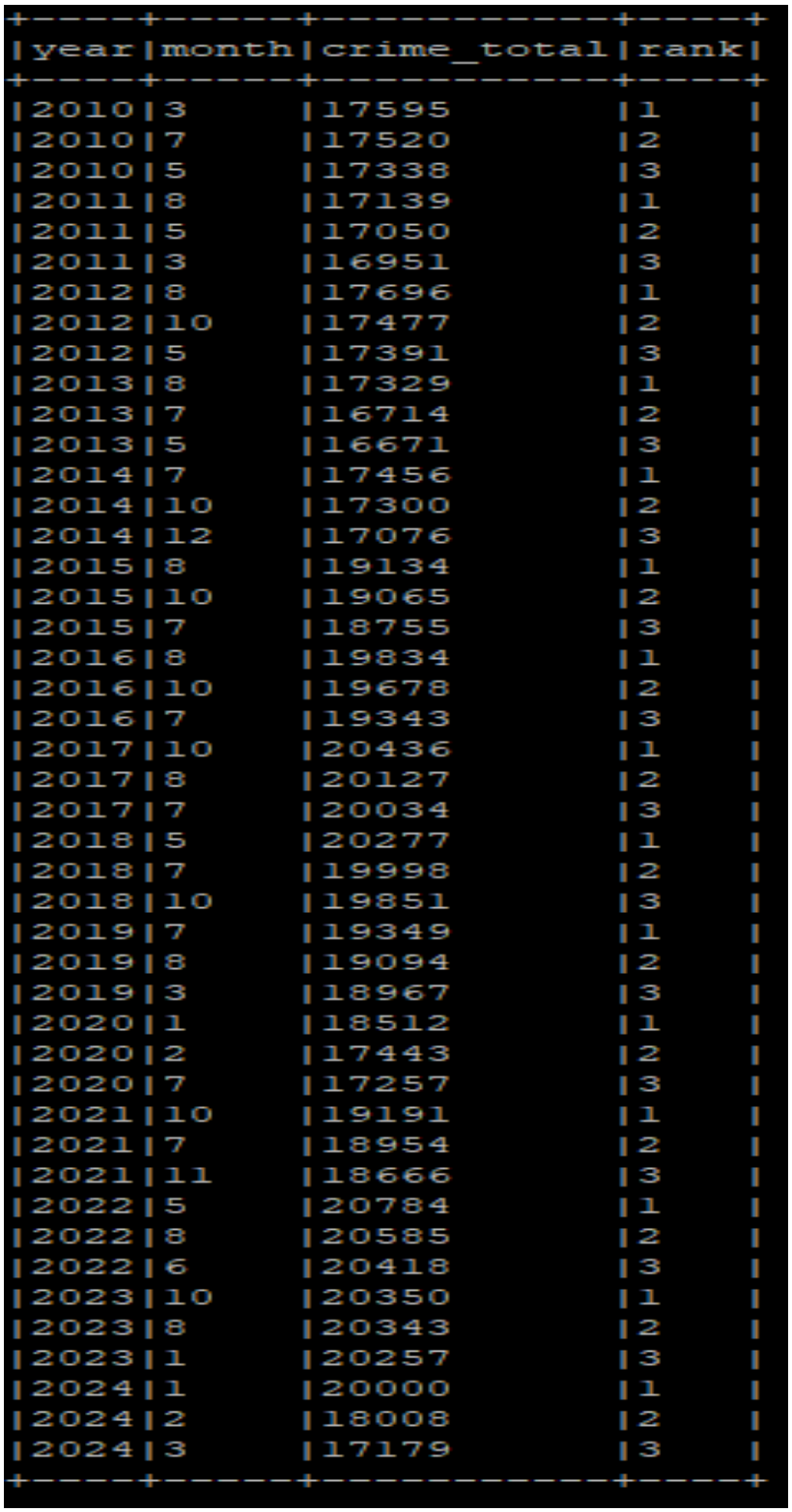
year	month	crime_total	rank
2010	3	17595	1
2010	7	17520	2
2010	5	17338	3
2011	8	17139	1
2011	5	17050	2
2011	3	16951	3
2012	8	17696	1
2012	10	17477	2
2012	5	17391	3
2013	8	17329	1
2013	7	16714	2
2013	5	16671	3
2014	7	17456	1
2014	10	17300	2
2014	12	17076	3
2015	8	19134	1
2015	10	19065	2
2015	7	18755	3
2016	8	19834	1
2016	10	19678	2
2016	7	19343	3
2017	10	20436	1
2017	8	20127	2
2017	7	20034	3
2018	5	20277	1
2018	7	19998	2
2018	10	19851	3
2019	7	19349	1
2019	8	19094	2
2019	3	18967	3
2020	1	18512	1
2020	2	17443	2
2020	7	17257	3
2021	10	19191	1
2021	7	18954	2
2021	11	18666	3
2022	5	20784	1
2022	8	20585	2
2022	6	20418	3
2023	10	20350	1
2023	8	20343	2
2023	1	20257	3
2024	1	20000	1
2024	2	18008	2
2024	3	17179	3

Σχήμα 2: Αποτέλεσμα υλοποίησης ερωτήματος 1 με χρήση SQL και csv αρχείων



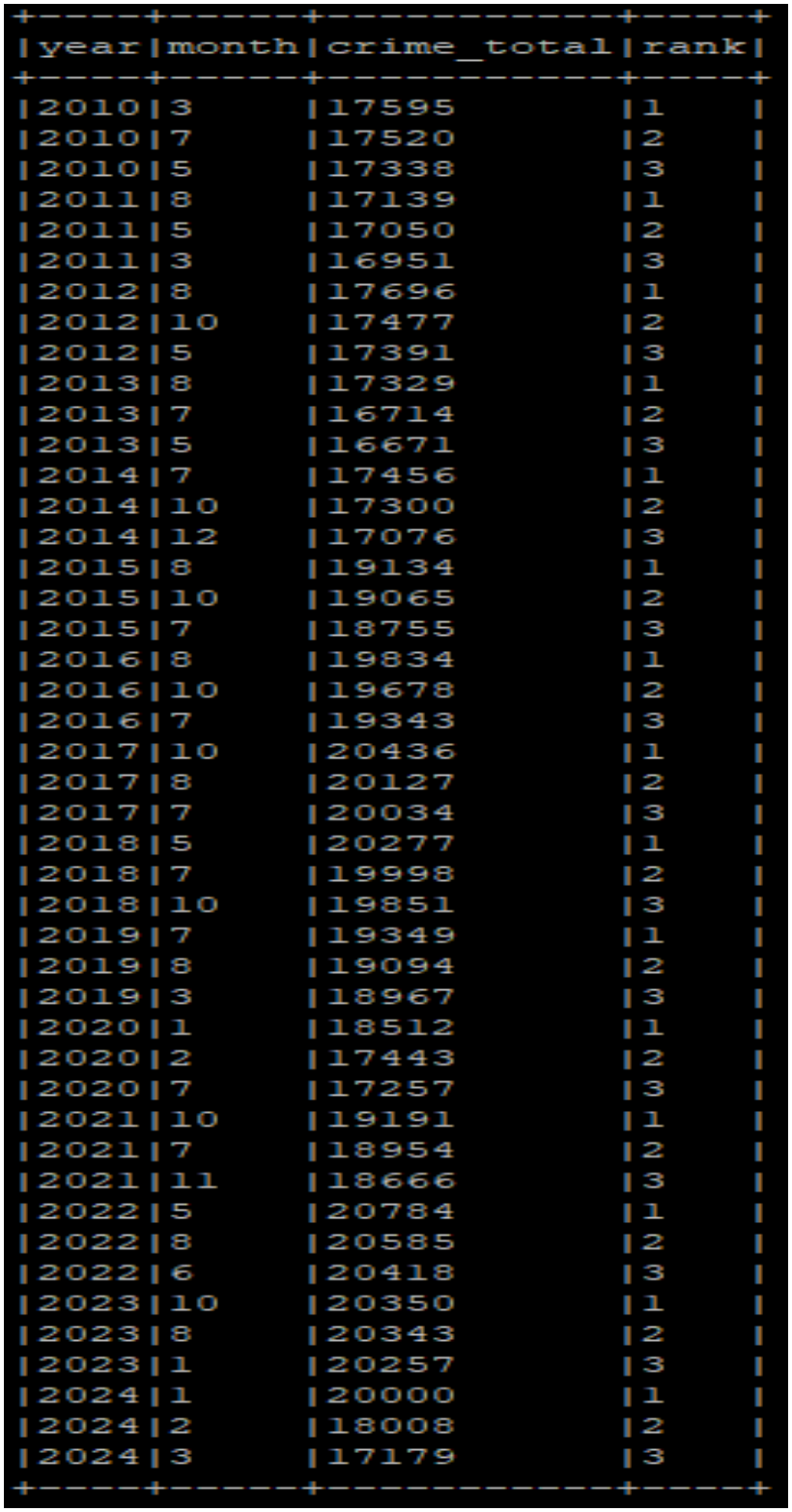
year	month	crime_total	rank
2010	3	17595	1
2010	7	17520	2
2010	5	17338	3
2011	8	17139	1
2011	5	17050	2
2011	3	16951	3
2012	8	17696	1
2012	10	17477	2
2012	5	17391	3
2013	8	17329	1
2013	7	16714	2
2013	5	16671	3
2014	7	17456	1
2014	10	17300	2
2014	12	17076	3
2015	8	19134	1
2015	10	19065	2
2015	7	18755	3
2016	8	19834	1
2016	10	19678	2
2016	7	19343	3
2017	10	20436	1
2017	8	20127	2
2017	7	20034	3
2018	5	20277	1
2018	7	19998	2
2018	10	19851	3
2019	7	19349	1
2019	8	19094	2
2019	3	18967	3
2020	1	18512	1
2020	2	17443	2
2020	7	17257	3
2021	10	19191	1
2021	7	18954	2
2021	11	18666	3
2022	5	20784	1
2022	8	20585	2
2022	6	20418	3
2023	10	20350	1
2023	8	20343	2
2023	1	20257	3
2024	1	20000	1
2024	2	18008	2
2024	3	17179	3

Σχήμα 3: Αποτέλεσμα υλοποίησης ερωτήματος 1 με χρήση SQL και parquet αρχείων



year	month	crime_total	rank
2010	3	17595	1
2010	7	17520	2
2010	5	17338	3
2011	8	17139	1
2011	5	17050	2
2011	3	16951	3
2012	8	17696	1
2012	10	17477	2
2012	5	17391	3
2013	8	17329	1
2013	7	16714	2
2013	5	16671	3
2014	7	17456	1
2014	10	17300	2
2014	12	17076	3
2015	8	19134	1
2015	10	19065	2
2015	7	18755	3
2016	8	19834	1
2016	10	19678	2
2016	7	19343	3
2017	10	20436	1
2017	8	20127	2
2017	7	20034	3
2018	5	20277	1
2018	7	19998	2
2018	10	19851	3
2019	7	19349	1
2019	8	19094	2
2019	3	18967	3
2020	1	18512	1
2020	2	17443	2
2020	7	17257	3
2021	10	19191	1
2021	7	18954	2
2021	11	18666	3
2022	5	20784	1
2022	8	20585	2
2022	6	20418	3
2023	10	20350	1
2023	8	20343	2
2023	1	20257	3
2024	1	20000	1
2024	2	18008	2
2024	3	17179	3

Σχήμα 4: Αποτέλεσμα υλοποίησης ερωτήματος 1 με χρήση dataframes και csv αρχείων



year	month	crime_total	rank
2010	3	17595	1
2010	7	17520	2
2010	5	17338	3
2011	8	17139	1
2011	5	17050	2
2011	3	16951	3
2012	8	17696	1
2012	10	17477	2
2012	5	17391	3
2013	8	17329	1
2013	7	16714	2
2013	5	16671	3
2014	7	17456	1
2014	10	17300	2
2014	12	17076	3
2015	8	19134	1
2015	10	19065	2
2015	7	18755	3
2016	8	19834	1
2016	10	19678	2
2016	7	19343	3
2017	10	20436	1
2017	8	20127	2
2017	7	20034	3
2018	5	20277	1
2018	7	19998	2
2018	10	19851	3
2019	7	19349	1
2019	8	19094	2
2019	3	18967	3
2020	1	18512	1
2020	2	17443	2
2020	7	17257	3
2021	10	19191	1
2021	7	18954	2
2021	11	18666	3
2022	5	20784	1
2022	8	20585	2
2022	6	20418	3
2023	10	20350	1
2023	8	20343	2
2023	1	20257	3
2024	1	20000	1
2024	2	18008	2
2024	3	17179	3

Σχήμα 5: Αποτέλεσμα υλοποίησης ερωτήματος 1 με χρήση dataframes και parquet αρχείων

Οι χρόνοι εκτέλεσης για τις διάφορες μεθόδους υλοποίησης του ερωτήματος παρουσιάζονται παρακάτω:

Table 1: Execution time for the different methods

Method	Execution time
SQL-CSV	1.4 min.
SQL-Parquet	46 sec.
Dataframe-CSV	1.3 min.
Dataframe-Parquet	40 sec.

Όπως είναι ευκόλα αντιληπτό ο χρόνος εκτέλεσης όταν χρησιμοποιούμε αρχεία εισόδου *parquet* είναι πολύ μικρότερος σε σύγκριση με τον χρόνο υλοποίησης με είσοδο αρχειών μορφής *csv*. Το γεγονός αυτό συμβαίνει για αρκετούς λόγους. Πρώτα απ'όλα, τα αρχεία *Parquet* χρησιμοποιούν μια δομή δεδομένων που είναι προσαρμοσμένη για ανάγνωση και εγγραφή από διαφορετικά εργαλεία επεξεργασίας δεδομένων, όπως το *Apache Spark* ή το *Apache Hive*. Αυτή η δομή επιτρέπει την αποθήκευση των δεδομένων σε μορφή στήλης, η οποία είναι πιο αποδοτική για πολλούς τύπους επεξεργασίας. Χρησιμοποιούν συμπίεση δεδομένων, που μπορεί να μειώσει το μέγεθος των αρχείων και αυτό έχει ως αποτέλεσμα να αποθηκεύουν περισσότερα δεδομένα στον ίδιο χώρο αποθήκευσης. Ακόμα, επειδή τα δεδομένα αποθηκεύονται σε συμπιεσμένη μορφή και με δομή στήλης, η ανάγνωση τους από εφαρμογές επεξεργασίας δεδομένων μπορεί να είναι πολύ πιο γρήγορη σε σχέση με τα αρχεία *CSV*.

4 Ζητούμενο 4

Οι χρόνοι εκτέλεσης για τις διάφορες υλοποιήσεις του ζητουμένου 4 φαίνονται παρακάτω.

Table 2: Execution Times for Different Methods

Method	Execution Time
RDD API	1.4 min
SQL API	52 s

Τα παραπάνω αποτελέσματα μπορούν να εξηγηθούν, αν διακρίνει κανείς τα χαρακτηριστικά των δύο APIs.

4.1 RDD API

- Χαμηλότερο Επίπεδο Αφαίρεσης: Η RDD είναι μια βασική δομή δεδομένων στο Spark, που αντιπροσωπεύει μια αμετάβλητη συλλογή αντικειμένων που μπορούν να επεξεργαστούν παράλληλα σε έναν συγκεκριμένο cluster.
- Χειροκίνητη Βελτιστοποίηση: Με τις RDD, οι προγραμματιστές έχουν περισσότερο έλεγχο στις εργασίες επεξεργασίας δεδομένων, αλλά πρέπει επίσης να ασχοληθούν με βελτιστοποιήσεις χαμηλού επιπέδου όπως η κατανομή δεδομένων.
- Έλλειψη Βελτιστοποίησης Ερωτήματος: Οι λειτουργίες RDD δεν επωφελούνται από τεχνικές υψηλού επιπέδου βελτιστοποίησης ερωτημάτων, που οδηγεί σε λιγότερο αποτελεσματικά σχέδια εκτέλεσης.

4.2 SQL API

- Υψηλότερο Επίπεδο Αφαίρεσης: Οι SQL-like APIs στο Spark, όπως τα DataFrame APIs, παρέχουν μια πιο καλή προσέγγιση στην επεξεργασία δεδομένων, επιτρέποντας στους προγραμματιστές να κάνουν τροποποιήσεις και διάφορες άλλες ενέργειες χρησιμοποιώντας SQL-likeσυντακτικό.
- Βελτιστοποίηση Ερωτήματος: Αυτά τα APIs εκμεταλλεύονται το Catalyst, τον βελτιστοποιητή ερωτημάτων του Spark, για να βελτιστοποιήσουν και να εκτελέσουν ερωτήματα αποτελεσματικά.

- Οφέλη στην Απόδοση: Λόγω των δυνατοτήτων βελτιστοποίησης ερωτημάτων και των υψηλότερων επιπέδων αφαίρεσης, τα APIs που βασίζονται σε SQL μπορούν συχνά να δημιουργήσουν πιο αποδοτικά σχέδια εκτέλεσης, οδηγώντας σε ταχύτερους χρόνους εκτέλεσης σε σύγκριση με υλοποιήσεις που βασίζονται σε RDD.

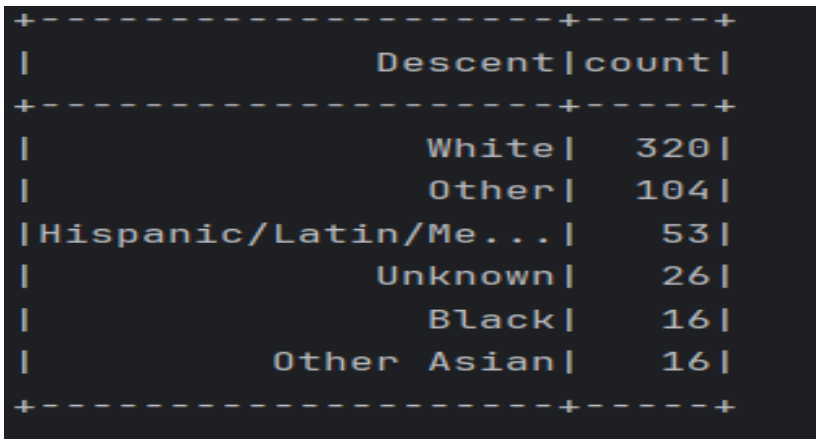
5 Ζητούμενο 5

Παρακάτω παρουσιάζονται οι χρόνοι εκτέλεσης των διαφόρων υλοποιήσεων του ερωτήματος, όπως αυτοί προέκυψαν για το ζητούμενο 5.

Table 3: Execution Times for Different Join Methods

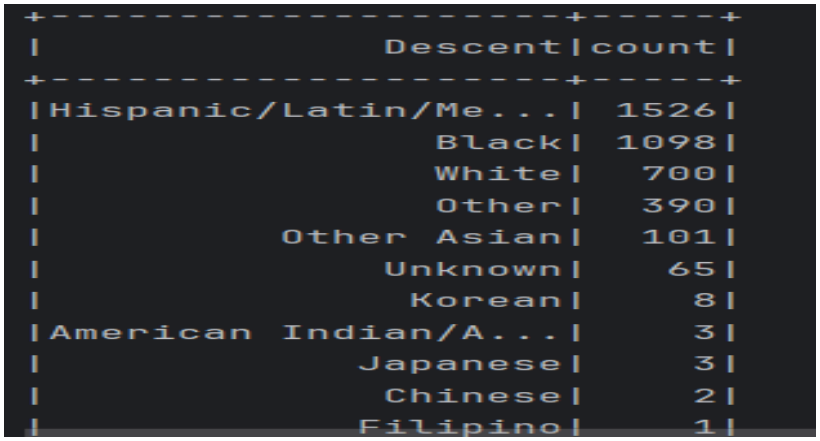
Join Method	**Execution Time**
Simple Join	2.0 min
Broadcast Join	2.0 min
Shuffle Hash Join	1.9 min

Κάθε υλοποιήση πραγματοποιήθηκε με ένα διαφορετικού είδους join. Το συμπέρασμα που μπορούμε να βγάλουμε με το συγκεκριμένο πίνακα είναι ότι και τα 3 είδη join εκτελούνται με σχεδόν ίδιο χρόνο, άλλα η υλοποίηση με Shuffle Hash Join είναι ελαφρώς ταχύτερη. Υπό αυτή τη συνθήκη θεωρούμε πιο απόδοτική υλοποίηση , εκείνη που χρησιμοποιήσε ως μέθοδο join το Shuffle Hash. Με βάση το τελευταίο συμπέρασμα και χρησιμοποιώντας γενικές γνώσεις, καταλήγουμε πως αν επρέπε να επιλέξουμε μία μέθοδο join, θα επιλέγαμε το Shuffle Hash Join. Το Shuffle Hash Join είναι κατάλληλο για μεγάλα datasets όπου απαιτείται η ανακατανομή (shuffling) των δεδομένων για να επιτευχθεί το join και μπορεί και να κατακερματίζει (hashes) τα δεδομένα και τα διανέμει στους κόμβους για πιο αποδοτικό join. Ένα ακόμη επιχείρημα που ενισχύει τη θέση μας, είναι ότι τα άλλα 2 είδη join δεν συνιστώνται για μεγάλα datasets.



Descent	count
White	320
Other	104
Hispanic/Latin/Me...	53
Unknown	26
Black	16
Other Asian	16

Σχήμα 6: Αποτέλσμα εκτέλεσης για εκείνους τους πολίτες με με το υψηλότερο εισόδημα



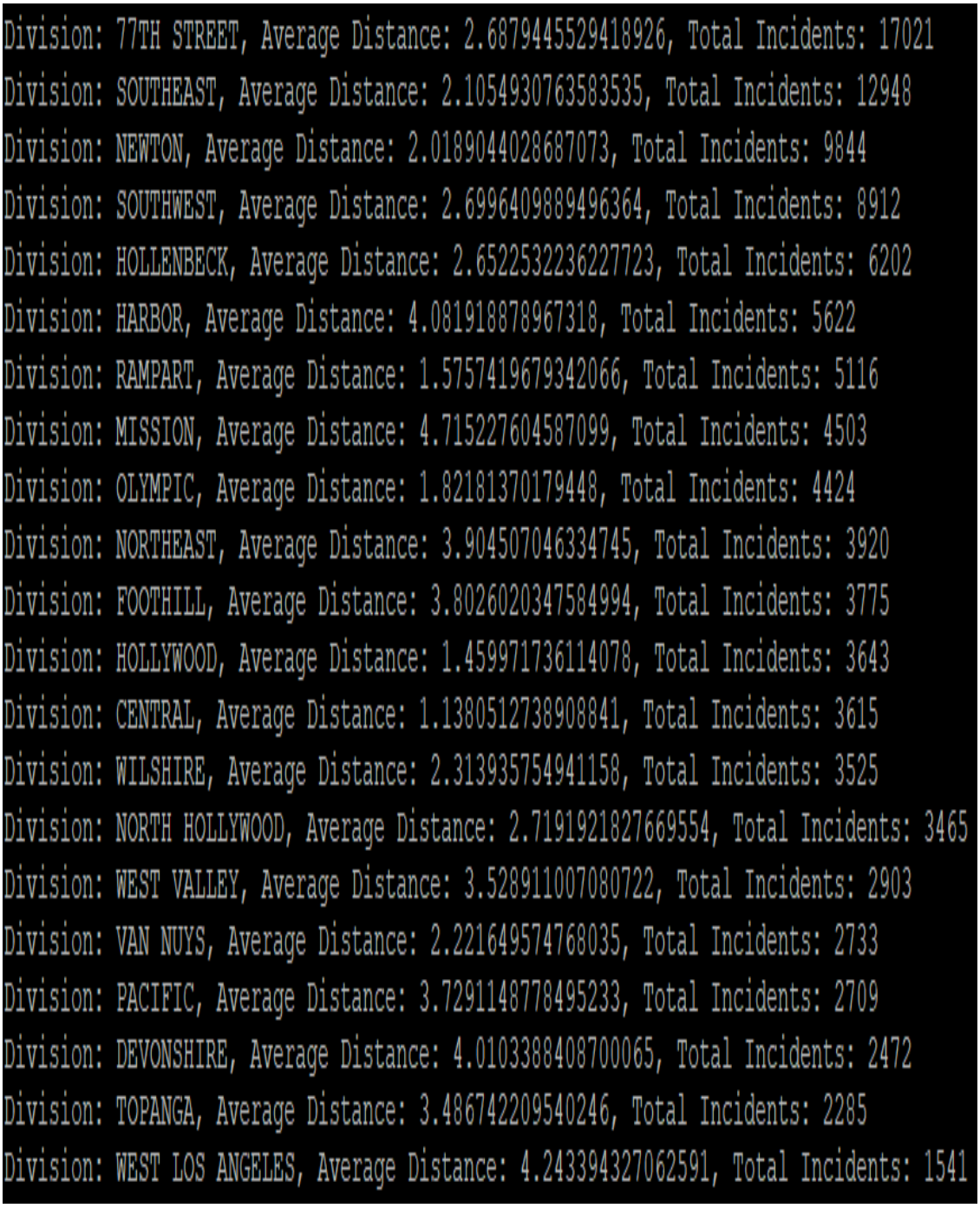
Descent	count
Hispanic/Latin/Me...	1526
Black	1098
White	700
Other	390
Other Asian	101
Unknown	65
Korean	8
American Indian/A...	3
Japanese	3
Chinese	2
Filipino	1

Σχήμα 7: Αποτέλσμα εκτέλεσης για εκείνους τους πολίτες με με το χαμηλότερο εισόδημα

6 Ζητούμενο 6

Παρακάτω παρουσιάζονται τα αποτελέσματα για τις απαιτούμενες υλοποιήσεις του ζητήματος, στην υποδεδειγμένη μορφή.

Σχήμα 8: Αποτέλεσμα της υλοποίησης με broadcast join.



Σχήμα 9: Αποτέλεσμα της υλοποίησης με repartition join.

7 Ζητούμενο 7

Παρακάτω εμφανίζεται το αποτέλεσμα της υλοποίησης για το εν λόγω ζητούμενο.

DIVISION	average_distance	incidents_total
77TH STREET	2.687863878979968	17021
SOUTHEAST	2.10536542436028	12948
NEWTON	2.01890408171001	9844
SOUTHWEST	2.69955690695839	8912
HOLLENBECK	2.652183094886917	6202
HARBOR	4.081888808832712	5622
RAMPART	1.5756271891244937	5116
MISSION	4.715336137664879	4503
OLYMPIC	1.8217951174232616	4424
NORTHEAST	3.9044950000219503	3920
FOOTHILL	3.8025931752182	3775
HOLLYWOOD	1.4599420127798626	3643
CENTRAL	1.1381342343584135	3615
WILSHIRE	2.313930122144256	3525
NORTH HOLLYWOOD	2.7191304762900135	3465
WEST VALLEY	3.5289325229810755	2903
VAN NUYS	2.221636501342287	2733
PACIFIC	3.729126540793401	2709
DEVONSHIRE	4.01035846098293	2472
TOPANGA	3.48689778486279	2285
WEST LOS ANGELES	4.243530477603231	1541

Σχήμα 10: Αποτέλεσμα της υλοποίησης με dataframes.