ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
Μάθημα: Προχωρημένα Θέματα Βάσεων Δεδομένων (9ο εξάμηνο) ακ. Έτος: 2024-2025
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ ΕΡΓΑΣΤΗΡΙΟ ΒΑΣΕΩΝ ΓΝΩΣΕΩΝ ΚΑΙ ΔΕΔΟΜΕΝΩΝ

Αναφορά Εξαμηνιαίας Εργασίας

Θέμα Εργασίας

Στην παρούσα εξαμηνιαία εργασία ζητείται ανάλυση σε (μεγάλα) σύνολα δεδομένων, εφαρμόζοντας επεξεργασία με τεχνικές που εφαρμόζονται σε data science projects. Τα εργαλεία που θα χρησιμοποιηθούν στα πλαίσια του project είναι τα Apache Hadoop (version>=3.0) και Apache Spark (version>=3.5). Καλείστε να χρησιμοποιήσετε τους πόρους στο ειδικά διαμορφωμένο περιβάλλον που σας έχει παραχωρηθεί στο AWS cloud. Συνοπτικά, ο σκοπός της εργασίας είναι:

• η εξοικείωση και ανάπτυξη των δεξιοτήτων των σπουδαστών στην εγκατάσταση και διαχείριση των κατανεμημένων συστημάτων Apache Spark και Apache Hadoop.
• Η χρήση σύγχρονων τεχνικών μέσω των API του Spark για την ανάλυση δεδομένων όγκου.
• Η κατανόηση των δυνατοτήτων και περιορισμών των εργαλείων αυτών σε σχέση με τους διαθέσιμους πόρους και τις ρυθμίσεις που έχουν επιλεγεί.

Group ID: 9
Βιτζηλαίος Γιώργος 03116672
Παπαδάκος Αλέξανδρος 03118860

GitHub:
ntua-el18860/Advanced-Database-Systems

ΠΕΡΙΕΧΟΜΕΝΑ

- **Εισαγωγή**

Η παρούσα αναφορά εκπονείται στα πλαίσια του μαθήματος "Προχωρημένα Θέματα Βάσεων Δεδομένων" της σχολής ΗΜΜΥ του ΕΜΠ. Αφορά την εξαμηνιαία εργασία του μαθήματος (το θέμα αναφέρεται στην πρώτη σελίδα της αναφοράς) και ουσιαστικά δίνει ολοκληρωμένες απαντήσεις στα 5 ζητούμενα που δίνονται στην εκφώνηση της εργασίας. Πρόκειται για 5 queries πάνω στα δεδομένα που αναλύονται και στο επόμενο κεφάλαιο. Για την υλοποίηση τους χρησιμοποιήθηκε η πλατφόρμα Amazon Web Services(AWS), όπου χρησιμοποιώντας το jupyterlab, τρέξαμε κώδικα σε python. Τα queries, όπως προδιαγράφεται και στις εκφωνήσεις, έγιναν μέσω του pyspark, χρησιμοποιώντας διάφορες υλοποιήσεις για τα APIs(DataFrame, RDD, SQL),για τα joins στους πίνακες, όπως και για τους πόρους. Αντίστοιχα χρησιμοποιείται η βιβλιοθήκη time ώστε να μετρήσουμε τον χρόνο που εκτελείται το κάθε query με την αντίστοιχη υλοποίηση και να εκτιμήσουμε την απόδοση της. Ακόμα, υπάρχει ιδιαίτερο ενδιαφέρον για την υλοποίηση queries που περιλαμβάνουν geospatial analytics. Εκεί χρησιμοποιούμε τη βιβλιοθήκη Apache Sedona (version 1.6.1), που έχει εγκατασταθεί στο περιβάλλον εργασίας μας (AWS). Περισσότερες πληροφορίες για την συγκεκριμένη βιβλιοθήκη, μπορούμε να βρούμε στο documentation και την ιστοσελίδα: https://sedona.apache.org/1.6.1/.

Στα παρακάτω κεφάλαια απαντώνται, όσο γίνεται πιο αναλυτικά, τα ζητούμενα τις εργασίας και γίνεται μία προσπάθεια εξαγωγής χρήσιμων συμπερασμάτων για τις αποδόσεις των τεχνικών που χρησιμοποιούμαι. Οι κώδικες δίνονται αναλυτικά στο github στο link που υπάρχει στην πρώτη σελίδα της αναφοράς, όπως και στα αντίστοιχα notebooks μας στο AWS.

Τέλος, θα θέλαμε να ευχαριστήσουμε τόσο τον καθηγητή μας Δημήτριο Τσουμάκο, όσο και τον υπεύθυνο του εργαστηρίου του μαθήματος Νικόλαο Χαλβαντζή, για την πολύτιμη βοήθεια τους στην εκπόνηση της παρούσας εργασίας και στην απάντηση οποιασδήποτε απορίας μας προέκυψε κατά την διάρκεια υλοποίησης της.

Οι φοιτητές:
Βιτζηλαίος Γιώργος
Αλέξανδρος Παπαδάκος
Ομάδα 09
18/01/2025

- **Δεδομένα**

Στην παράγραφο αυτή θα παρουσιαστούν τα δεδομένα που θα κληθήκαμε να χρησιμοποιήσουμε στα πλαίσια της εξαμηνιαίας εργασίας. Πρόκειται για δημοσίως διαθέσιμα και δωρεάν σύνολα δεδομένων που έχουν συλλεχθεί από διαφορετικές πηγές.

Βασικό data-set: Los Angeles Crime Data

Το βασικό σύνολο δεδομένων που θα χρησιμοποιηθεί στην εργασία προέρχεται από το δημόσιο αποθετήριο δεδομένων της κυβέρνησης των Ηνωμένων Πολιτειών της Αμερικής1. Συγκεκριμένα, περιλαμβάνει δεδομένα καταγραφής εγκλημάτων για το Los Angeles από το 2010 μέχρι σήμερα. Τα δεδομένα είναι διαθέσιμα σε csv file format στους παρακάτω συνδέσμους:

• https://data.lacity.org/Public-Safety/Crime-Data-from-2010-to-2019/63jg-8b9z
• https://data.lacity.org/Public-Safety/Crime-Data-from-2020-to-Present/2nrs-mtv8

Στους ίδιους συνδέσμους παρέχονται περιγραφές για κάθε ένα από τα 28 πεδία των δεδομένων.

Δευτερεύοντα data-sets

Συμπληρωματικά με τα παραπάνω δεδομένα, θα χρησιμοποιηθεί μια σειρά δεδομένων μικρότερου όγκου τα οποία επίσης είναι διαθέσιμα σε δημόσια αποθετήρια ή πηγές : 2010 Census Blocks (Los Angeles County): Ένα σύνολο δεδομένων που παρουσιάζει απογραφικά στοιχεία που αφορούν στην Κομητεία του Los Angeles για το έτος 2010 σε geojson format. Συνοδεύεται από αρχείο με περιγραφές των πεδίων του (2010_Census_Blocks_fields.csv). Είναι διαθέσιμο στον παρακάτω σύνδεσμο:

• https://data.lacounty.gov/maps/lacounty::2010-census-blocks

Median Household Income by Zip Code (Los Angeles County): Ένα ακόμα μικρό σύνολο δεδομένων που περιέχει δεδομένα σχετικά με το μέσο εισόδημα ανά νοικοκυριό και ταχυδρομικό κώδικα (ZIP Code) στην Κομητεία του Los Angeles. Για διευκόλυνση, τα δεδομένα έχουν συλλεχθεί και αποθηκευθεί σε csv file format. Τα συγκεκριμένο σύνολο δεδομένων παράχθηκε με βάση τα αποτελέσματα της απογραφής του έτους 2015 και είναι διαθέσιμα στον παρακάτω σύνδεσμο:

• http://www.laalmanac.com/employment/em12c_2015.php

LA Police Stations: Μικρό σύνολο δεδομένων που περιέχει δεδομένα σχετικά με την τοποθεσία των 21 αστυνομικών τμημάτων που βρίσκονται στην πόλη του Los Angeles. Τα συγκεκριμένα δεδομένα προέρχονται από δημόσιο αποθετήριο δεδομένων του δήμου του Los Angeles και είναι διαθέσιμα σε csv file format στον παρακάτω σύνδεσμο:

• https://geohub.lacity.org/datasets/lahub::lapd-police-stations/explore

Race and Ethnicity codes: Ένα μικρό σύνολο δεδομένων που περιέχει τις πλήρες περιγραφές που αντιστοιχούν στην κωδικοποίηση του φυλετικού προφίλ που χρησιμοποιείται στο βασικό σύνολο δεδομένων.

- **Ζητούμενα Εργασίας**
- **Query 1**

Να ταξινομηθούν, σε φθίνουσα σειρά, οι ηλικιακές ομάδες των θυμάτων σε περιστατικά που περιλαμβάνουν οποιαδήποτε μορφή "βαριάς σωματικής βλάβης".

Θεωρείστε τις εξής ηλικιακές ομάδες:
• Παιδιά: < 18
• Νεαροί ενήλικοι: 18 – 24
• Ενήλικοι: 25 – 64
• Ηλικιωμένοι: >64

Ζητούμενο 1: Να υλοποιηθεί το Query 1 χρησιμοποιώντας τα DataFrame και RDD APIs. Να εκτελέσετε και τις δύο υλοποιήσεις με 4 Spark executors. Υπάρχει διαφορά στην επίδοση μεταξύ των δύο APIs; Αιτιολογήσετε την απάντησή σας.

Σημείωση: Ως εγκλήματα που περιλαμβάνουν οποιαδήποτε μορφή "βαριάς σωματικής βλάβης" θεωρούμε όλα εκείνα τα περιστατικά που περιέχουν τον όρο "aggravated assault" στη σχετική περιγραφή.

Απάντηση:

Τα αποτελέσματα του κώδικα μας δίνονται αναλυτικά στον github, όπου σε ειδικά αρχεία parqeut δίνονται όλα τα δεδομένα ταξινομημένα σε φθίνουσα σειρά, ανά ηλικιακή ομάδα. Στον κώδικα που έχουμε επίσης στο github έχουμε αφήσει την περίπτωση όπου εμφανίζονται ταξινομημένα σε φθίνουσα σειρά, μόνο τα πρώτα 20 δεδομένα ανά ηλικιακή ομάδα, για εξοικονόμηση χώρου και πιο βέλτιστης ανάγνωσης του κώδικα.

Ο κώδικας εκτελέστηκε με δύο υλοποιήσεις. Η μία με DataFrame API και η δεύτερη με RDD API. Και στις δύο περιπτώσεις με 4 Spark executors. Αυτό που παρατηρούμε είναι ότι **υπάρχει διαφορά στην επίδοση μεταξύ των δύο APIs.** Πιο συγκεκριμένα ο κώδικας που τρέχει με **DataFrame υλοποιεί το query μέσα σε 14.28 δευτερόλεπτα,** ενώ ο κώδικας που τρέχει με **RDD, υλοποιεί το query μέσα σε 35.24 δευτερόλεπτα.**

Το παραπάνω μοιάζει αρκετά λογικό μιας και το **DataFrame API** διαχειρίζεται πιο εύκολα δεδομένα που είναι δομημένα και σε πίνακες, ενώ την ίδια στιγμή παρέχει μηχανισμούς για την βέλτιστη υλοποίηση ενός query. Για παράδειγμα, στο συγκεκριμένο ερώτημα, θα επιλέξει κατευθείαν τις στήλες των θυμάτων με "βαριά σωματική βλάβη" και απλά θα ταξινομήσει τον πίνακα σε φθίνουσα σειρά για κάθε ηλικιακή ομάδα. Το **RDD API** θα σπάσει την κάθε σειρά του πίνακα δεδομένων μας σε ξεχωριστό κόμβο και θα προσπαθήσει να ταξινομήσει μετά τα δεδομένα, υλοποίηση που προφανώς είναι πιο χρονοβόρα.

- **Query 2**

Να βρεθούν, για κάθε έτος, τα 3 Αστυνομικά Τμήματα με το υψηλότερο ποσοστό κλεισμένων (περατωμένων) υποθέσεων. Να τυπωθούν το έτος, τα ονόματα (τοποθεσίες) των τμημάτων, τα ποσοστά τους καθώς και οι αριθμοί του ranking τους στην ετήσια κατάταξη. Τα αποτελέσματα να δοθούν σε σειρά αύξουσα ως προς το έτος και το ranking (δείτε παράδειγμα στον Πίνακα 2).

Ζητούμενο 2.α): Να υλοποιηθεί το Query 2 χρησιμοποιώντας τα DataFrame και SQL APIs. Να αναφέρετε και να συγκρίνετε τους χρόνους εκτέλεσης μεταξύ των δύο υλοποιήσεων.

Ζητούμενο 2.β): Να γράψετε κώδικα Spark που μετατρέπει το κυρίως data set σε parquet2 file format και αποθηκεύει ένα μοναδικό .parquet αρχείο στο S3 bucket της ομάδας σας. Επιλέξτε μία από τις δύο υλοποιήσεις του υπό ερωτήματος α) (DataFrame ή SQL) και συγκρίνετε τους χρόνους εκτέλεσης της εφαρμογής σας όταν τα δεδομένα εισάγονται σαν .csv και σαν .parquet.

Απάντηση:

α)

```
|year|precinct   |closed_case_rate |rank|
+----+-----------+-----------------+----+
|2010|Rampart    |32.94735585531813|1   |
|2010|Olympic    |31.96270619172842|2   |
|2010|Harbor     |29.63203463203463|3   |
|2011|Olympic    |35.21216768916155|1   |
|2011|Rampart    |32.51177963030083|2   |
|2011|Harbor     |28.65220520201501|3   |
|2012|Olympic    |34.41481831052383|1   |
|2012|Rampart    |32.94641810294290|2   |
|2012|Harbor     |29.81513327601032|3   |
|2013|Olympic    |33.52812271731191|1   |
|2013|Rampart    |32.08287360549222|2   |
|2013|Harbor     |29.16422459266206|3   |
|2014|Van Nuys   |31.80567315834039|1   |
|2014|West Valley|31.31198995605775|2   |
|2014|Mission    |31.16279069767442|3   |
|2015|Van Nuys   |32.64134698172773|1   |
|2015|West Valley|30.27597402597403|2   |
|2015|Mission    |30.17946067838016|3   |
|2016|Van Nuys   |31.88075572011773|1   |
|2016|West Valley|31.54798761609907|2   |
|2016|Foothill   |29.87029184335246|3   |
|2017|Van Nuys   |32.02034211742950|1   |
|2017|Mission    |31.03892518634398|2   |
|2017|Foothill   |30.46922608165753|3   |
|2018|Foothill   |30.70895065507530|1   |
|2018|Mission    |30.69066147859922|2   |
|2018|Van Nuys   |29.07868573051794|3   |
|2019|West Valley|30.77447195094254|1   |
|2019|Mission    |30.74851911685514|2   |
|2019|Foothill   |29.53842186694172|3   |
|2020|West Valley|31.31156039398652|1   |
|2020|Mission    |30.16882799119158|2   |
|2020|Harbor     |30.11040092969204|3   |
|2021|Mission    |30.62377210216110|1   |
|2021|West Valley|28.81677128982589|2   |
|2021|Foothill   |28.31351196446482|3   |
|2022|West Valley|26.66407917717834|1   |
|2022|Harbor     |26.30313626926019|2   |
|2022|Topanga    |26.21687255833597|3   |
|2023|Foothill   |26.80668785408318|1   |
|2023|Topanga    |26.48504711183941|2   |
|2023|Mission    |25.81423401688782|3   |
|2024|N Hollywood|19.60671738194345|1   |
|2024|Foothill   |18.44836533471718|2   |
|2024|77th Street|17.40947075208914|3   |
+----+-----------+-----------------+----+

Time taken: 15.49 seconds
```

```
year   precinct            closed_case_rate     #
------------------------------------------------------------
2010   Rampart             32.94736             1
2010   Olympic             31.96271             2
2010   Harbor              29.63203             3
2011   Olympic             35.21217             1
2011   Rampart             32.51178             2
2011   Harbor              28.65221             3
2012   Olympic             34.41482             1
2012   Rampart             32.94642             2
2012   Harbor              29.81513             3
2013   Olympic             33.52812             1
2013   Rampart             32.08287             2
2013   Harbor              29.16422             3
2014   Van Nuys            31.80567             1
2014   West Valley         31.31199             2
2014   Mission             31.16279             3
2015   Van Nuys            32.64135             1
2015   West Valley         30.27597             2
2015   Mission             30.17946             3
2016   Van Nuys            31.88076             1
2016   West Valley         31.54799             2
2016   Foothill            29.87029             3
2017   Van Nuys            32.02034             1
2017   Mission             31.03893             2
2017   Foothill            30.46923             3
2018   Foothill            30.70895             1
2018   Mission             30.69066             2
2018   Van Nuys            29.07869             3
2019   West Valley         30.77447             1
2019   Mission             30.74852             2
2019   Foothill            29.53842             3
2020   West Valley         31.14489             1
2020   Mission             30.38778             2
2020   Harbor              29.88066             3
2021   Mission             30.91391             1
2021   West Valley         28.87750             2
2021   Foothill            28.46479             3
2022   West Valley         26.64366             1
2022   Harbor              26.33406             2
2022   Topanga             26.27340             3
2023   Foothill            26.82160             1
2023   Topanga             26.40781             2
2023   Mission             25.94120             3
2024   N Hollywood         19.51498             1
2024   Foothill            18.53183             2
2024   77th Street         17.34914             3
Time taken: 7.01 seconds
```

Αριστερά βλέπουμε τα αποτελέσματα του ερωτήματος χρησιμοποιώντας SQL API, ενώ από τα δεξιά χρησιμοποιώντας DataFrame API. **Βλέπουμε ότι και εδώ το DataFrame API τρέχει πιο γρήγορα** και είναι λογικό μιάς και χρησιμοποιεί μηχανισμούς για να κάνει τα queries βέλτιστα σε σχέση με την απλή SQL.

β) Γράφουμε κώδικα που μετατρέπει σε parqeut τα δεδομένα μας, ενώ επιλέγουμε την υλοποιήση με DataFrame. Ξανά τρέχουμε τώρα τις δύο περιπτώσεις, όπου στην μία διαβάζουμε τα δεδομένα από το parqeut αρχείο και στην άλλη από το csv. Παρατηρούμε ότι **στην πρώτη περίπτωση το πρόγραμμα τρέχει σε 11.58 δευτερόλεπτα, ενώ στην δεύτερη σε 30.47.** Λογικό αποτέλεσμα μιας και τα parqeut αρχεία αποθηκεύουν τα δεδομένα σε στήλες όχι σε γραμμές όπως τα csv, συμπιέζουν τα δεδομένα τους ώστε να μειώνεται σε μεγάλο βαθμό ο όγκος που μεταφέρεται από τον δίσκο στην μνήμη, μπορούν να εφαρμόζουν ερωτήματα πάνω στα δεδομένα κατά την ανάγνωση και παράλληλα. Όλα αυτά τα καθιστούν καταλληλότερα στην αποθήκευση δεδομένων και σαφώς πιο γρήγορα στην επεξεργασία τους.

- ## Query 3

Χρησιμοποιώντας ως αναφορά τα δεδομένα της απογραφής 2010 για τον πληθυσμό και εκείνα της απογραφής του 2015 για το εισόδημα ανά νοικοκυριό, να υπολογίσετε για κάθε περιοχή του Los Angeles τα παρακάτω: Το μέσο ετήσιο εισόδημα ανά άτομο και την αναλογία συνολικού αριθμού εγκλημάτων ανά άτομο. Τα αποτελέσματα να συγκεντρωθούν σε ένα πίνακα.

Ζητούμενο 3: Να υλοποιηθεί το Query 3 χρησιμοποιώντας DataFrame ή SQL API. Χρησιμοποιήστε τις μεθόδους hint & explain για να βρείτε ποιες στρατηγικές join χρησιμοποιεί ο catalyst optimizer. Πειραματιστείτε αναγκάζοντας το Spark να χρησιμοποιήσει διαφορετικές στρατηγικές (μεταξύ των BROADCAST, MERGE, SHUFFLE_HASH, SHUFFLE_REPLICATE_NL) και σχολιάστε τα αποτελέσματα που παρατηρείτε. Ποιά (ή ποιές) από τις διαθέσιμες στρατηγικές join του Spark είναι καταλληλότερη(ες) και γιατί;

Σημείωση: Θεωρήστε ότι οι περιοχές του Los Angeles ορίζονται από τη στήλη COMM του 2010 Census Blocks.

Απάντηση: Παρακάτω παραθέτουμε αρχικά τα αποτελέσματα από το μέσο ετήσιο εισόδημα ανά άτομο και έπειτα την αναλογία συνολικού αριθμού εγκλημάτων ανά άτομο. Τα παρακάτω αποτελέσματα τα βρήκαμε ακολουθώντας τα εξής βήματα:

1. Πήραμε τα δεδομένα από το «2010_Census_Blocks.geojson» και από το «LA_income_2015.csv».
2. Κάνουμε join τους 2 πίνακες για να παρουμε για τις περιοχές του LA τα εισοδήματα και τους πληθυσμούς.
3. Παίρνουμε τα εισοδήματα ανά νοικοκυριό, τα πολλαπλασιάζουμε με τα νοικοκυριά ανά περιοχή, βρίσκουμε το συνολικό εισόδημα ανά περιοχή και το διαιρούμε με τον συνολικό πληθυσμό για να βρούμε το Income per Individual.

Community	Total Population	Total Households	Estimated Median Income per Household	Income per Individual
Culver City	77766	34982	$75,913.50	$34,148.68
Pico Rivera	62942	17109	$55,758.00	$15,156.23
Malibu	12645	6864	$123,681.00	$67,136.92
Hacienda Heights	53594	16524	$78,000.00	$24,048.81
Montebello	62500	19768	$45,898.00	$14,516.99
Hawaiian Gardens	14254	3703	$37,543.00	$9,753.17
Westlake Village	16540	6768	$104,382.50	$42,712.26
Carson	183428	52452	$74,351.50	$21,261.12
Glendale	1150314	457614	$66,520.33	$26,462.89
Signal Hill	11016	4389	$65,935.00	$26,269.85
Glendora	101388	35964	$76,010.00	$26,962.00
Gardena	176487	64416	$49,137.67	$17,934.76
Claremont	35348	12306	$89,161.00	$31,040.38
Norwalk	105549	28083	$59,756.00	$15,899.04
Monterey Park	120538	41700	$53,341.50	$18,453.44
Beverly Hills	102327	49182	$102,447.00	$49,239.68
Paramount	54098	14571	$45,792.00	$12,333.82
La Cañada Flintridge	20246	7089	$156,933.00	$54,949.03
Altadena	42777	15947	$86,310.00	$32,175.83
Ladera Heights	6498	2867	$84,099.00	$37,105.55
La Mirada	48527	15092	$78,373.00	$24,374.17
Artesia	16522	4697	$60,749.00	$17,270.19
Santa Monica	269208	152736	$93,137.33	$52,841.76
South Gate	94396	24160	$43,552.00	$11,146.83
San Marino	13147	4477	$134,286.00	$45,728.94
Hermosa Beach	19506	10162	$111,187.00	$57,924.86
Acton	8134	3075	$86,944.00	$32,868.55
East Los Angeles	126492	32199	$37,916.00	$9,651.66
Diamond Bar	55544	18455	$90,522.00	$30,076.76
Alhambra	166178	61830	$54,360.50	$20,225.96
Torrance	581752	233508	$74,483.75	$29,896.85
San Gabriel	79436	26474	$61,311.00	$20,433.40
Covina	128078	42082	$63,711.50	$20,933.39
Temple City	35558	12117	$63,410.00	$21,608.05
Sierra Madre	10917	5113	$92,384.00	$43,268.24
Lomita	20256	8412	$56,103.00	$23,298.70
Bellflower	76616	24897	$48,823.00	$15,865.44
Rosemead	53764	14805	$46,729.00	$12,867.77
Cerritos	49587	16003	$89,720.00	$28,954.95
Monrovia	40446	15681	$70,307.00	$27,258.17
El Monte	227226	58224	$38,070.00	$9,754.99
Bradbury	1292	592	$90,000.00	$41,238.39
Pomona	302012	81800	$47,705.00	$12,920.91
Whittier	177744	61738	$68,485.50	$23,787.91
Lynwood	69772	15277	$43,231.00	$9,465.69
Arcadia	127866	46792	$77,771.50	$28,460.14

Αντίστοιχα, για το παρακάτω query, εργαζόμαστε ως εξής:

1. Κάνουμε join τους πίνακες, «2010_Census_Blocks.geojson» και τα δεδομένα από τα εγκλήματα την τελευταία δεκαετία. Κατανέμουμε τον αριθμό των εγκλημάτων ανά περιοχή LA μέσω της βιβλιοθήκης sedona.
2. Διαιρούμε τον αριθμό των εγκλημάτων ανά περιοχή με τον αριθμό των κατοίκων ανά περιοχή και βρίσκουμε το Ratio_of_Crime_Per_Person.

```
+--------------------+----------------+-------------+--------------------------+
|Community           |Total Population|Sum_of_Crimes|Ratio_of_Crimes_Per_Person|
+--------------------+----------------+-------------+--------------------------+
|Culver City         |77766           |2780         |0.03574827045238279       |
|Pico Rivera         |62942           |2            |3.177528518318452E-5      |
|Malibu              |12645           |1            |7.908264136022143E-5      |
|Hacienda Heights    |53594           |NULL         |NULL                      |
|Montebello          |62500           |6            |9.6E-5                    |
|Hawaiian Gardens    |14254           |NULL         |NULL                      |
|Westlake Village    |16540           |2            |1.2091898428053205E-4     |
|Carson              |183428          |862          |0.004699391586889679      |
|Glendale            |1150314         |816          |7.093715281218867E-4      |
|Signal Hill         |11016           |2            |1.8155410312273057E-4     |
|Glendora            |101388          |2            |1.9726200339290646E-5     |
|Gardena             |176487          |1311         |0.007428309167247446      |
|Claremont           |35348           |6            |1.697408622835804E-4      |
|Norwalk             |105549          |5            |4.7371363063600794E-5     |
|Monterey Park       |120538          |10           |8.296138976920142E-5      |
|Beverly Hills       |102327          |4059         |0.03966695007182855       |
|Paramount           |54098           |31           |5.730341232577914E-4      |
|La Cañada Flintridge|20246           |1            |4.9392472587177715E-5     |
|Altadena            |42777           |1            |2.3377048413867265E-5     |
|Ladera Heights      |6498            |945          |0.14542936288088643       |
|La Mirada           |48527           |1            |2.0607084715725267E-5     |
|Artesia             |16522           |NULL         |NULL                      |
|Santa Monica        |269208          |2385         |0.008859320673977         |
|South Gate          |94396           |2            |2.1187338446544345E-5     |
|San Marino          |13147           |NULL         |NULL                      |
|Hermosa Beach       |19506           |18           |9.227929867733005E-4      |
|Acton               |8134            |1            |1.2294074256208508E-4     |
|East Los Angeles    |126492          |440          |0.0034784808525440344     |
|Diamond Bar         |55544           |1            |1.8003744778914014E-5     |
|Alhambra            |166178          |174          |0.0010470700092671714     |
|Torrance            |581752          |2680         |0.004606774020544837      |
|San Gabriel         |79436           |NULL         |NULL                      |
|Covina              |128078          |2            |1.5615484314246007E-5     |
|Temple City         |35558           |NULL         |NULL                      |
|Sierra Madre        |10917           |NULL         |NULL                      |
|Lomita              |20256           |76           |0.0037519747235387046     |
|Bellflower          |76616           |3            |3.9156311997494E-5        |
|Rosemead            |53764           |1            |1.8599806562011755E-5     |
|Cerritos            |49587           |5            |1.008328795853752E-4      |
|Monrovia            |40446           |1            |2.472432378974435E-5      |
|El Monte            |227226          |8            |3.520723860825786E-5      |
|Bradbury            |1292            |NULL         |NULL                      |
|Pomona              |302012          |14           |4.635577394275724E-5      |
|Whittier            |177744          |4            |2.2504275812404356E-5     |
|Lynwood             |69772           |28           |4.01307114601846E-4       |
|Arcadia             |127866          |4            |3.1282749127993365E-5     |
```

Για τους χρόνους των join έχουμε τα εξής:

Join Strategy	**Task_1**	**Task_2**
Broadcast	14.76 sec	88.68 sec
Merge	11.43 sec	91.60 sec
Shuffle_Hash	10.19 sec	86.56 sec
Shuffle_Replicate_NL	13.05 sec	158.13 sec

Για τα join επισημαίνουμε τα εξής:

- Το **Broadcast Join** είναι μία από τις πιο αποδοτικές στρατηγικές όταν ένας από τους δύο πίνακες είναι μικρός και μπορεί να μεταδοθεί (broadcast) σε όλους τους worker nodes. Το Spark αντιγράφει τον μικρότερο πίνακα σε όλους τους εκτελεστές (executors), επιτρέποντας στους κόμβους να εκτελέσουν το join τοπικά, χωρίς ανάγκη μεταφοράς δεδομένων.Ο μικρότερος πίνακας αναπαράγεται (broadcast) σε όλους τους worker nodes. Κάθε κόμβος εκτελεί το join τοπικά με τον μεγαλύτερο πίνακα που έχει κατανεμηθεί. Τα πλεονεκτήματα του είναι ότι είναι εξαιρετικά αποδοτικό όταν ένας πίνακας είναι πολύ μικρός, μειώνει την ανάγκη μεταφοράς δεδομένων μέσω του

δικτύου, λειτουργεί πολύ καλά σε μεγάλα κατανεμημένα συστήματα. Αντίστοιχα, τα μειονεκτήματα του είναι ότι αν ο μικρότερος πίνακας είναι πολύ μεγάλος, μπορεί να οδηγήσει σε out of memory errors (OOM) και απαιτεί επαρκή μνήμη σε κάθε εκτελεστή για την αποθήκευση του μικρού πίνακα.

- Το **Merge Join** είναι η προτιμώμενη στρατηγική όταν και οι δύο πίνακες είναι ήδη ταξινομημένοι (sorted) ως προς το join key. Αν οι πίνακες δεν είναι ταξινομημένοι, το Spark πρώτα τους ταξινομεί με βάση το join key. Εφόσον οι πίνακες είναι ταξινομημένοι, το join εκτελείται με διαδοχική σύγκριση (merge-like scan). Μερικά πλεονεκτήματα είναι ότι είναι εξαιρετικά αποδοτικό αν τα δεδομένα είναι ήδη ταξινομημένα, λειτουργεί καλά για πολύ μεγάλα σύνολα δεδομένων, χρησιμοποιεί διαδοχική σάρωση (sequential scan), η οποία είναι αποδοτική για μεγάλες κλίμακες δεδομένων.Μερικά μειονεκτήματα είναι ότι αν οι πίνακες δεν είναι ταξινομημένοι, τότε το κόστος ταξινόμησης μπορεί να είναι υψηλό, όπως επίσης ότι αν και οι δύο πίνακες είναι πολύ μεγάλοι, μπορεί να χρειαστεί επιπλέον μνήμη και αποθήκευση για το sorting.
- Το **Shuffle Hash Join** είναι μια δημοφιλής τεχνική join που βασίζεται στην κατανομή (shuffling) των δεδομένων μεταξύ των worker nodes και στη χρήση δομών δεδομένων hash για την εκτέλεση του join. Οι πίνακες κατανέμονται (shuffle) στους κόμβους με βάση το join key. Κάθε worker node χτίζει έναν πίνακα hash για τον έναν από τους δύο πίνακες. Ο δεύτερος πίνακας διατρέχεται και αναζητείται το αντίστοιχο κλειδί στον hash table. Μερικά πλεονεκτήματα της συγκεκριμένης τεχνικής είναι ότι λειτουργεί πολύ καλά όταν ένας από τους δύο πίνακες μπορεί να χωρέσει στη μνήμη, αποδίδει καλύτερα από το Sort-Merge Join όταν τα δεδομένα δεν είναι ταξινομημένα, έχει χαμηλότερο κόστος από το Sort-Merge Join, ειδικά σε μικρότερα δεδομένα. Αντίστοιχα, μειονεκτήματα είναι αν ο ένας πίνακας δεν χωρά στη μνήμη, η απόδοση μπορεί να μειωθεί δραματικά, όπως επίσης περιλαμβάνει υψηλό shuffle overhead, κάτι που μπορεί να επιβαρύνει το δίκτυο.
- Το **Shuffle Replicate NL** είναι η πιο γενική και ακριβή σε πόρους μέθοδος join. Χρησιμοποιείται συνήθως όταν δεν υπάρχει κάποιο κατάλληλο index ή όταν τα δεδομένα δεν μπορούν να ταξινομηθούν ή κατανεμηθούν αποδοτικά. Όλοι οι κόμβοι του cluster αντιγράφουν (replicate) τα δεδομένα και εκτελούν nested loop join, όπου κάθε εγγραφή του πρώτου πίνακα συγκρίνεται με κάθε εγγραφή του δεύτερου. Μερικά πλεονεκτήματα είναι ότι λειτουργεί σε όλες τις περιπτώσεις, ακόμη και όταν δεν υπάρχουν index ή sorting keys, επίσης δεν απαιτεί ταξινομημένα δεδομένα. Κάποια μειονεκτήματα είναι η πιο αργή και δαπανηρή στρατηγική, καθώς το κόστος είναι O(N*M), αντιγράφει τα δεδομένα σε όλους τους worker nodes, καταναλώνοντας πολύ μνήμη και bandwidth και τέλος χρησιμοποιείται μόνο όταν όλες οι άλλες στρατηγικές αποτυγχάνουν.

- **Query 4**

Να βρεθεί το φυλετικό προφίλ των καταγεγραμμένων θυμάτων εγκλημάτων (Vict Descent) στο Los Angeles για το έτος 2015 στις 3 περιοχές με το υψηλότερο κατά κεφαλήν εισόδημα. Να γίνει το ίδιο για τις 3 περιοχές με το χαμηλότερο εισόδημα. Να χρησιμοποιήσετε την αντιστοίχιση των κωδικών καταγωγής με την πλήρη περιγραφή από το σύνολο δεδομενων Race and Ethnicity codes. Τα αποτελέσματα να τυπωθούν σε δύο ξεχωριστούς πίνακες από το υψηλότερο στο χαμηλότερο αριθμό θυμάτων ανά φυλετικό γκρουπ (δείτε παράδειγμα αποτελέσματος στον Πίνακα 3).

Ζητούμενο 4: Να υλοποιηθεί το Query 4 χρησιμοποιώντας το DataFrame ή SQL API. Να εκτελέσετε την υλοποίησή σας εφαρμόζοντας κλιμάκωση στο σύνολο των υπολογιστικών πόρων που θα χρησιμοποιήσετε: Συγκεκριμένα, καλείστε να εκτελέστε την υλοποίησή σας σε 2 executors με τα ακόλουθα configurations:
• 1 core/2 GB memory
• 2 cores/4GB memory
• 4 cores/8GB memory
Σχολιάστε τα αποτελέσματα.

Σημείωση: Κάποιες εγγραφές του βασικού συνόλου δεδομένων λανθασμένα αναφέρονται στο Null Island (0,0). Θα πρέπει να φιλτραριστούν και να μη λαμβάνονται υπόψη στον υπολογισμό των αποστάσεων.

Απάντηση: Αρχικοποιούμε τις βιβλιοθήκες που θα χρησιμοποιήσουμε για να διαβάσουμε τις κατάλληλες μορφές αρχείων που χρειάζονται για το συγκεκριμένο ερώτημα.
Φιλτράρουμε τα δεδομένα μας και δημιουργούμε το flattened_df για να ταιριάζει με τους αντίστοιχους T.K και παράλληλα δημιουργούμε μια στήλη ST_POINT για την χωρική επεξεργασία των δεδομένων μας.

Στη συνέχεια για την χωρική ένωση χρησιμοποιούμε την εντολή ST_Contains για τη σύγκριση των γεωμετριών μας, δηλαδή για να εξετάσουμε αν υπάρχουν εγκλήματα μέσα στο γεωμετρικό χώρο που μας ενδιαφέρει την εκάστοτε φορά. Ομαδοποιούμε τα δεδομένα μας βάση της στήλης Vict_descent και υπολογίζουμε τις αντίστοιχες συχνότητες για τις 3 πλουσιότερες και τις 3 φτωχότερες περιοχές. Στη συνέχεια εκτελούμε την ίδια διαδικασία για τα διαφορετικά configurations που μας ζητούνται . Οι αντίστοιχοι χρόνοι φαίνονται στα εκάστοτε notebook (4, 4.1 , 4.2). Τα αποτελέσματα είναι τα εξής :

Για τις 3 πλουσιότερες περιοχές :

```
+--------------------+---+
|   Vict Descent Full|  #|
+--------------------+---+
|               Black| 56|
|            Japanese|  1|
|Hispanic/Latin/Me...|127|
|            Filipino|  2|
|               Other|128|
|             Unknown| 90|
|               White|470|
|         Other Asian| 24|
+--------------------+---+
```

Για τις 3 φτωχότερες περιοχές :

```
+--------------------+---+
|   Vict Descent Full|  #|
+--------------------+---+
|               Black|  5|
|Hispanic/Latin/Me...|  8|
|               Other| 15|
|             Unknown|  8|
|               White| 57|
|         Other Asian|  2|
+--------------------+---+
```

- **Query 5**

Να υπολογιστεί, ανά αστυνομικό τμήμα, ο αριθμός εγκλημάτων που έλαβαν χώρα πλησιέστερα σε αυτό, καθώς και η μέση απόστασή του από τις τοποθεσίες όπου σημειώθηκαν τα συγκεκριμένα περιστατικά. Τα αποτελέσματα να εμφανιστούν ταξινομημένα κατά αριθμό περιστατικών, με φθίνουσα σειρά.

Ζητούμενο 5: Να υλοποιηθεί το Query 5 χρησιμοποιώντας το DataFrame ή SQL API. Να εκτελέσετε την υλοποίησή σας χρησιμοποιώντας συνολικούς πόρους 8 cores και 16GB μνήμης με τα παρακάτω configurations:

• 2 executors × 4 cores/8GB memory
• 4 executors × 2 cores/4GB memory
• 8 executors × 1 core/2 GB memory
Σχολιάστε τα αποτελέσματα.

Απάντηση:
Εργαστήκαμε ως εξής για την εξαγωγή των παρακάτω αποτελεσμάτων:

1. Αρχικά, πήραμε τα δεδομένα από το LA_Police_Stations.csv για τα αστυνομικά τμήματα, όπως και τα δεδομένα από τα εγκλήματα από τα main csvs.
2. Στην συνέχεια αθροίσαμε τα εγκλήματα που αντιστοιχούν σε κάθε αστυνομικό τμήμα. Μέσω της βιβλιοθήκης sedona παίρνουμε τα στίγματα πάνω στο χάρτη από το που έγιναν τα εγκλήματα, και τα κατανέμουμε με βάση ποια ανήκουν μέσα στα πολύγωνα του κάθε αστυνομικού τμήματος(περιοχή ευθύνης).
3. Πάλι μέσω της βιβλιοθήκης sedona βρίσκουμε τις αποστάσεις και τα σορτάρουμε σε φθίνουσα σειρά.

```
+----------------+------------------+------+
|division        |average_distance  |#     |
+----------------+------------------+------+
|77th street     |14667.404665088128|206981|
|southwest       |11919.183392928315|192367|
|pacific         |23217.837897333302|171166|
|central         |19800.237298414453|166946|
|north hollywood |16209.757573536073|164710|
|southeast       |18117.102515811494|161256|
|hollywood       |34031.54247813007 |151053|
|newton          |12954.564101781074|148886|
|olympic         |16840.905966885035|145135|
|mission         |20202.014726011737|143777|
|northeast       |12730.983942391795|142833|
|van nuys        |14156.99316736504 |142327|
|topanga         |9801.350722735635 |138708|
|devonshire      |18727.924888877016|138044|
|wilshire        |18742.94184561195 |136374|
|rampart         |16044.949875248454|136104|
|west los angeles|13881.883428494097|134369|
|harbor          |15430.634370288273|133031|
|west valley     |11522.972959349752|131585|
|hollenbeck      |18947.554681490175|114665|
|foothill        |15852.825436448838|113020|
+----------------+------------------+------+
```

Η απόδοση των προγραμμάτων ήταν η εξής:

- 2 executors × 4 cores/8GB memory : 25.28 δευτερόλεπτα

Αυτή η διαμόρφωση επιτυγχάνει την καλύτερη απόδοση, καθώς διαθέτει επαρκή μνήμη **(8GB) και 4 cores ανά executor**, επιτρέποντας τη βελτιστοποιημένη παράλληλη εκτέλεση με μειωμένο overhead από το shuffling. Οι λιγότεροι αλλά ισχυρότεροι executors μπορούν

να εκτελέσουν μεγαλύτερα tasks ανά core, μειώνοντας την ανάγκη επικοινωνίας μεταξύ των worker nodes.

- 4 executors × 2 cores/4GB memory : 28.33 δευτερόλεπτα

Αυτή η διαμόρφωση έχει περισσότερους executors αλλά με λιγότερους πόρους (cores & memory) ανά executor. Η απόδοση είναι χειρότερη από την πρώτη περίπτωση, πιθανότατα λόγω του αυξημένου overhead επικοινωνίας μεταξύ των executors, καθώς η μνήμη είναι χαμηλότερη.

- 8 executors × 1 core/2 GB memory : 27.58 δευτερόλεπτα

Αν και υπάρχουν περισσότεροι executors, ο κάθε executor έχει μόνο 1 core και 2GB μνήμης, κάτι που σημαίνει ότι δεν μπορεί να εκτελέσει πολλές εργασίες ταυτόχρονα. Επιπλέον, ο αυξημένος αριθμός executors οδηγεί σε περισσότερο shuffling και overhead στη διαχείριση των εργασιών, γεγονός που εξηγεί την χαμηλότερη απόδοση σε σχέση με την πρώτη περίπτωση.