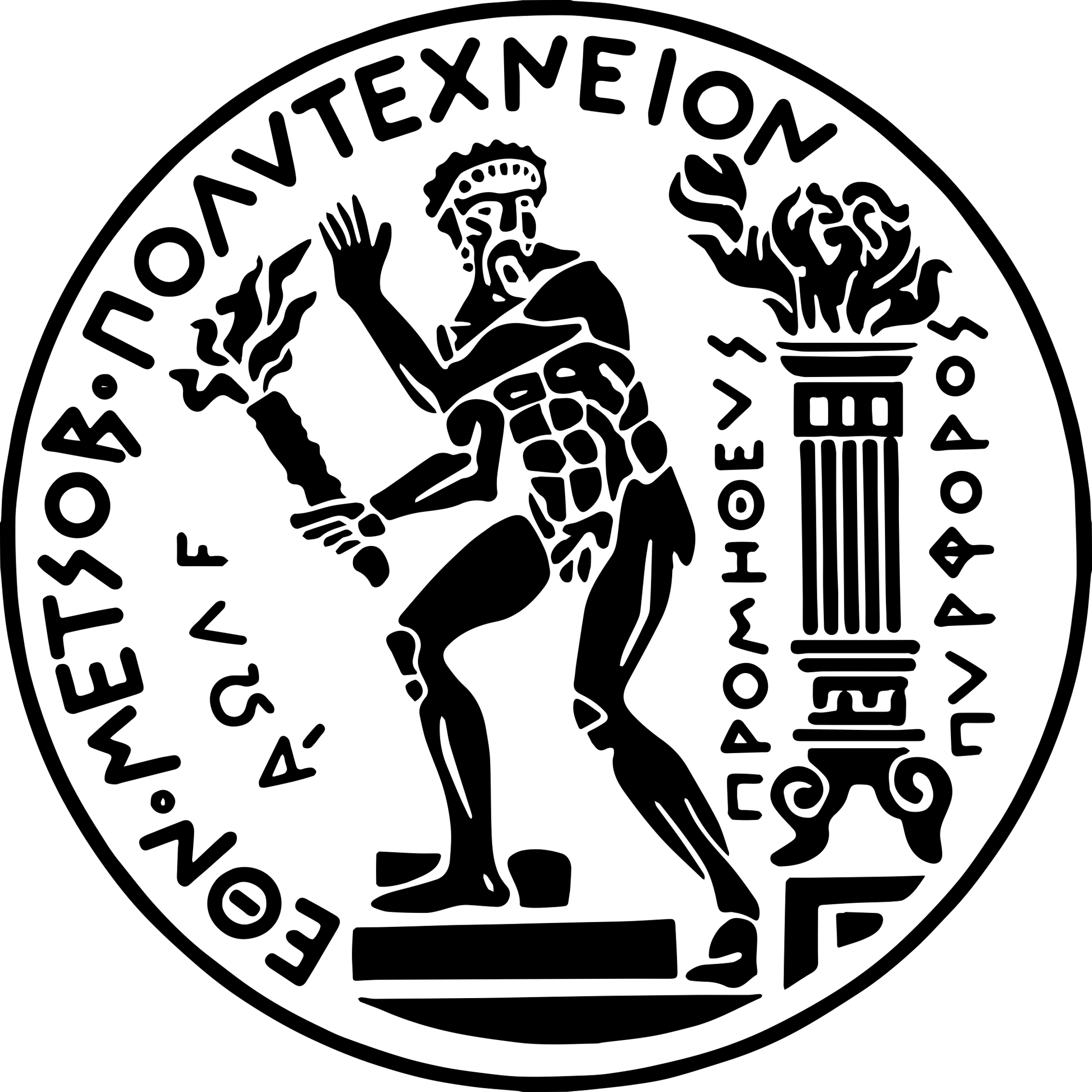
ΕΞΑΜΗΝΙΑΙΑ ΕΡΓΑΣΙΑ ΠΡΟΧΩΡΗΜΕΝΑ ΘΕΜΑΤΑ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ



Σπουδαστής: Γιαννακόπουλος Στέφανος (03120829) 9ο Εξάμηνο

Σπουδάστρια: Κασιούμη Παρασκευή (03120122) 9ο Εξάμηνο

Αρ.Ομάδας: 37

Github Repository: <https://github.com/stefanosGiannakopoulos/AdvancedDB_SemesterProject2024-2025>

***Όλα τα αποτελέσματα για τα queries καθώς και τα ζητούμενα αρχεία, βρίσκονται στο public github repository που καταγράφεται παραπάνω.***

ΣΕ ΟΛΕΣ ΤΙΣ ΧΡΟΝΟΜΕΤΡΗΣΕΙΣ ΛΑΒΑΜΕ ΥΠΟΨΙΝ ΤΟ LAZY EVALUATION

Queries:

1)

Στρατηγική:

α) DataFrame

Ουσιαστικά, απλά ενώσαμε τα 2 csv για να έχουμε ένα ενιαίο dataframe στο οποίο κρατήσαμε μόνο τις περιπτώσεις όπου είχαμε βαριά σωματική βλάβη. Ύστερα, συναρτήσει της ηλικίας των θυμάτων τα ταξινομήσαμε στις ακόλουθες κατηγορίες (Children, Young Adults, Adults, Elderly, Invalid Age), όπου ομαδοποιήσαμε ανά ομάδα (group by ), κάναμε καταμέτρηση των περιστατικών (περάσαμε aggregation count στο group) και κάναμε sorting σε φθίνουσα σειρά.

β) RDD

Κάναμε ό,τι και πριν απλά εδώ χρησιμοποιήσαμε map reduce. Ειδικότερα:

**Map**: Μετατροπή κάθε εγγραφής σε ζεύγος (key, value), με το key να είναι η ηλικιακή κατηγορία και το value να είναι 1.

**Reduce**: Ομαδοποίηση όλων των ζευγών με το ίδιο key και άθροισμα των value για κάθε κατηγορία.

**Sort**: Ταξινόμηση των αποτελεσμάτων κατά φθίνουσα σειρά.

Ζητούμενο:

Για το DataFrame API απαιτήθηκαν 6.2 s για την εύρεση του τελικού αποτελέσματος, ενώ για το RDD API απαιτήθηκαν 15 s . Η διαφορά στον χρόνο εκτέλεσης μεταξύ του **DataFrame API** (6.2 δευτερόλεπτα) και του **RDD API** (15 δευτερόλεπτα) οφείλεται κυρίως στη βελτιστοποίηση που προσφέρει το DataFrame API μέσω του **Catalyst Optimizer**, ο οποίος εφαρμόζει τεχνικές όπως προώθηση φίλτρων, συγχώνευση λειτουργιών και δημιουργία αποδοτικών σχεδίων εκτέλεσης. Το DataFrame API χρησιμοποιεί αποδοτική αναπαράσταση δεδομένων (Internal Row) και εκτελεί λειτουργίες σε bytecode, ενώ το RDD API χειρίζεται γενικά αντικείμενα, που είναι πιο αργά και καταναλώνουν περισσότερη μνήμη. Επιπλέον, το DataFrame API περιορίζει την κίνηση δεδομένων στο δίκτυο και εκμεταλλεύεται καλύτερα τη συμπίεση και τη διαχείριση μνήμης. Αντίθετα, το RDD API προκαλεί περισσότερες λειτουργίες αναδιάταξης δεδομένων (shuffling) και δεν βελτιστοποιεί αυτόματα τις λειτουργίες, γεγονός που οδηγεί σε μεγαλύτερο χρόνο εκτέλεσης.

2)

Στρατηγική:

α) i) DataFrame

Ζητούμενα:

Το **SQL API** είναι συχνά ταχύτερο από το **DataFrame API** επειδή αξιοποιεί άμεσα τον **Catalyst Optimizer**, που εφαρμόζει βελτιστοποιήσεις πιο αποτελεσματικά. Η δηλωτική φύση του SQL API επιτρέπει στο Spark να παραλείψει περιττά ενδιάμεσα βήματα, ενώ το DataFrame API απαιτεί περισσότερη μετάφραση σε λογικό σχέδιο. Επιπλέον, το SQL API εκμεταλλεύεται βελτιστοποιήσεις, όπως προώθηση φίλτρων, βελτιστοποίηση joins και χρήση μεταδεδομένων από τον Catalog Manager, για αποδοτικότερη εκτέλεση. Ειδικά για σύνθετα ερωτήματα, το SQL API είναι καλύτερο στη δημιουργία αποδοτικών σχεδίων εκτέλεσης χωρίς επιβάρυνση από ενδιάμεσες λειτουργίες

5)