Project 2 Report

Big Data Analysis and Management Systems

Φίλιππος Δουραχαλής, f3312205

Γενική μεθοδολογία:

Για την υλοποίηση του project χρησιμοποιήθηκε η γλώσσα προγραμματισμού python καθώς παρέχει API για την εύκολη συγγραφή εφαρμογών σε Spark, ενώ περιέχει και πολλές ακόμα βιβλιοθήκες για ανάλυση δεδομένων όπως είναι η Pandas και η matplotlib η οποία μας επιτρέπει να δημιουργήσουμε χωρίς κόπο πληθώρα γραφημάτων για την αναπαράσταση των δεδομένων.

Ως λειτουργικό σύστημα χρησιμοποιήθηκε το distribution Linux Mint, το οποίο βασίζεται στα Ubuntu και το οποίο είχε εγκατασταθεί σε ένα VM.

Πριν την εκτέλεση ξεκινάμε τον master καθώς και έναν worker για την εκτέλεση των tasks.

Για κάθε ερώτημα τα βήματα που ακολουθήθηκαν -γενικά- είναι τα ακόλουθα:

1. Δημιουργία DataFrame μέσω της ανάγνωσης του αντίστοιχου csv αρχείου (εφόσον δεν έχει αναγνωσθεί ήδη σε προηγούμενο ερώτημα)
2. Δημιουργία όψης μέσω της μεθόδου createOrReplaceTempView() ώστε να μπορούμε να εκτελέσουμε native SQL ερωτήματα επί του πίνακα δεδομένων
3. Εκτέλεση ενός ή περισσότερων SQL ερωτημάτων για την παραγωγή των επιθυμητών (ενδιάμεσων ή μη) αποτελεσμάτων
4. (Προαιρετικά) Εκτέλεση επιπρόσθετων μετασχηματισμών στο DataFrame που έχουν επιστρέψει τα SQL ερωτήματα ή join των ενδιάμεσων αποτελεσμάτων για την παραγωγή του τελικού αποτελέσματος
5. Μετατροπή του DataFrame σε Pandas DataFrame (**σημαντικό!**) για την αποθήκευση του σε ένα νέο csv αρχείο

Για την εκτέλεση του script που παράγει τις αναφορές που ζητούνται χρησιμοποιήθηκε η εντολή "*spark-submit script.py*" που υποβάλει την εφαρμογή για κατανεμημένη επεξεργασία

Εμπόδια και δυσκολίες:

Κατά τη διάρκεια της ανάπτυξης εμφανίστηκαν αρκετά προβλήματα σχετικά με την υλοποίηση των ερωτημάτων, την χρήση των μεθόδων που παρείχε το API ή bugs που αναφέρονταν στην αποθήκευση των csv αρχείων. Αυτά τα προβλήματα παρουσιάζονται στη συνέχεια.

Καταρχάς προέκυψε η ανάγκη για έλεγχο της ύπαρξης των αρχείων αποτελέσματος ώστε να γίνει αντικατάστασή τους σε περίπτωση που έχουν δημιουργηθεί ήδη και να παίρνουμε κάθε φορά ένα έγκυρο αποτέλεσμα.

Ερωτήματα 1 & 2: Στα συγκεκριμένα ερωτήματα δεν αντιμετωπιστήκαν ιδιαίτερα εμπόδια όσον αφορά την παραγωγή των αποτελεσμάτων των ίδιων των ερωτημάτων. Ωστόσο σημαντικό εμπόδιο εδώ αποτέλεσε η επιλογή των κατάλληλων μεθόδων για την σχεδίαση των γραφημάτων. Για να γίνει αυτό υπήρχαν δύο επιλογές. Είτε να χρησιμοποιηθεί απευθείας η βιβλιοθήκη matplotlib της Python που είναι ο πλέον ενδεδειγμένος τρόπος για την παραγωγή γραφημάτων για ανάλυση δεδομένων και παρέχει πλήθος διαφορετικών μεθόδων για την εξατομίκευση τους, ή να χρησιμοποιηθούν οι έτοιμες μέθοδοι που παρέχει το Pnadas, το οποίο κάνει χρήση της βιβλιοθήκης matplotlib, αλλά είναι ευκολότερη στη χρήση καθώς τα γραφήματα μπορούν να παραχθούν και να αποθηκευτούν σε δύο μόλις γραμμές.

Αρχικά ακολουθήθηκε η δεύτερη προσέγγιση, η οποία ήταν ένας άμεσος τρόπος να δημιουργηθούν τα γραφήματα. Ωστόσο σύντομα προέκυψε το πρόβλημα της παραμετροποίησης των γραφημάτων, καθώς το Pandas μπορεί μεν να παράγει πιο εύκολα τα γραφήματα, αλλά οποιαδήποτε αλλαγή επί αυτών απαιτεί πιο πολύπλοκους χειρισμούς.

Για τον λόγο αυτό στη συνέχεια χρησιμοποιήθηκε η matplotlib η οποία απαιτεί κάποιες επιπλέον ενέργειες (π.χ. εκκαθάριση του πρώτου γραφήματος πρωτού κατασκευάσουμε το δεύτερο), αλλά επιτρέπει πολύ ευκολότερη παραμετροποίηση τους προκειμένου να παραχθεί ένα καλαίσθητο αποτέλεσμα.

Ερώτημα 3

Εδώ η προσέγγιση της λύσης ήταν σχετικά απλά. Η λογική της υλοποίησης έχει ως εξής:

Εφόσον στον πίνακα orderProducts (που έχει παραχθεί από το DataFrame του csv order_products) υπάρχει το πεδίο reordered που παίρνει τιμές {0,1}, αρκεί να ομαδοποιήσουμε τα τμήματα και τα προϊόντα και για κάθε group να πάρουμε το άθροισμα του reordered. Αν ένα προϊόν δεν έχει παραγγελθεί ξανά στο παρελθόν, το άθροισμα του θα είναι μηδέν (παντού reordered = 0), άρα το επιλέγουμε για το τελικό αποτέλεσμα.

Αρχικά είχε επιλεχθεί μια προσέγγιση όπου εντοπίζονταν πρώτα όλα τα προϊόντα που **έχουν** παραγγελθεί στο παρελθόν (reordered = 1) με ξεχωριστό ερώτημα και στη συνέχεια αποκλείονταν με την εντολή NOT IN ώστε να επιλεχθούν τα προϊόντα που δεν ανήκαν σε αυτόν τον ενδιάμεσο πίνακα. Ωστόσο αυτή η υλοποίηση δεν ήταν αποδοτική, επομένως προτιμήθηκε η προηγούμενη.

Ένα σημαντικό πρόβλημα που αντιμετωπίστηκε σε αυτό το στάδιο ήταν η απευθείας αποθήκευση του DataFrame ως csv. Μετά την αποθήκευση και κατά το άνοιγμα του csv με το πρόγραμμα LibreOffice Calc, προκειμένου να γίνει επαλήθευση του αποτελέσματος, ορισμένες γραμμές έλλειπαν χωρίς προφανή λόγο (δηλαδή ενώ στο terminal όπου εκτελούταν το Spark εμφανιζόταν το σωστό πλήθος γραμμών, στο LibreOffice Calc εμφανίζονταν περίπου 700 λιγότερες). Αυτό προκάλεσε σημαντικούς προβληαμτισμούς σχετικά με το που βρισκόταν το πρόβλημα και δαπανήθηκε αρκετός χρόνος στο να εντοπισθεί, παρότι όλες οι εναλλακτικές λύσεις (SQL ερωτήματα) παρήγαγαν το ίδιο αποτέλεσμα κατά την εκτέλεση. Εν τέλει διαπιστώθηκε ότι αυτό οφειλόταν στην αποθήκευση του ίδιου του Spark DataFrame. Μετατρέποντας το DataFrame σε Pandas, δημιουργούταν σωστά το αρχείο csv και οι γραμμές που εκτυπώνονταν στο terminal είχαν πλήρη αντιστοιχία με αυτές που εμφάνιζε το LibreOffice Calc. Δεν γνωρίζω εάν το συγκεκριμένο πρόβλημα εν τέλει οφειλόταν σε κάποιο bug του LibreOffice ή του τρόπου με τον οποίο γίνεται η αποθήκευση των αρχείων στο Spark, καθώς δεν έγινε περαιτέρω διερεύνησή του.

Ερώτημα 4:

Σε αυτό το ερώτημα ζητείται το όνομα του προϊόντος που έχει παραγγελθεί περισσότερες φορές από κάθε τμήμα. Ωστόσο προέκυψαν αμφιβολίες στο τι συμβαίνει εάν δύο προϊόντα έχουν παραγγελθεί τις ίδιες (μέγιστες) φορές. Η SQL καθώς και το Spark παρέχει μεθόδους με τις οποίες μπορούμε αφού έχουμε βρει και ταξινομήσει το πλήθος των παραγγελιών για κάθε προϊόν ενός τμήματος, να κρατήσουμε μόνο την πρώτη εγγραφή για κάθε κατηγορία τμήματος (δηλαδή το μέγιστό του), όπως η dropDuplicates() που εφαρμόζεται στο DataFrame του αποτελέσματος του αρχικού SQL ερωτήματος.

Ωστόσο με αυτές τις μεθόδους δεν έχουμε κανέναν τρόπο να πάρουμε το δεύτερο προϊόν που έχει ισοβαθμία με το πρώτο. Ακόμα δυσκολότερο γίνεται το έργο μας αν θέλουμε όλα τα προϊόντα που έχουν το ίδιο μέγιστο για ένα τμήμα.

Για τον λόγο αυτό έγινε η παραδοχή ότι θέλουμε να κρατήσουμε όλα τα προϊόντα που έχουν ισοβαθμία, προκείμενου να μην χάσουμε (ενδεχομένως χρήσιμη) πληροφορία. Έτσι αφού πάρουμε το DataFrame που περιέχει για κάθε τμήμα και κάθε προϊόν το πλήθος των φορών που έχει παραγγελθεί ξανά με ένα SQL ερώτημα, επιλέγουμε επαναληπτικά για ένα τμήμα μόνο την πρώτη εγγραφή, αλλά ελέγχουμε παράλληλα αν για το ίδιο τμήμα υπάρχει και άλλο προϊόν με το ίδιο μέγιστο, ώστε να το προσθέσουμε και την εγγραφή για εκείνο.

Ερώτημα 5:

Φτάνοντας στο τελευταίο ερώτημα, το πιο σημαντικό εμπόδιο που προέκυψε ήταν το πως θα παραχθεί το ποσοστό στο τελικό αποτέλεσμα. Η πιο άμεση λύση ήταν η εξής:

Αρχικά υπολογίζουμε το πλήθος των προϊόντων ενός διαδρόμου που έχουν τοποθετηθεί έστω μια φορά στο καλάθι κάποιας παραγγελίας. Αυτό αποτελεί τον αριθμητή μας. Στη συνέχεια υπολογίζεται ο παρονομαστής του ποσοστού, δηλαδή το συνολικό πλήθος των προϊόντων ενός διαδρόμου. Έχοντας αυτά τα δύο ξεχωριστά DataFrames, μένει να τα ενώσουμε και να βγάλουμε το τελικό ποσοστό. Ο πιο εύκολος τρόπος να γίνει αυτό, ώστε να μην χρησιμοποιηθεί ξανά ένα τρίτο SQL ερώτημα, ήταν η χρήση της συνάρτησης join() πάνω στο κοινό πεδίο των DataFrames (aisle) και στη συνέχεια μετατροπή τους σε RDD προκειμένου να γίνει χρήση της συνάρτησης map η οποία διατρέχει κάθε γραμμή του RDD, διαιρεί το πεδίο αριθμητή με τον παρονομαστή και πολλαπλασιάζει το αποτέλεσμα επί 100 για να προκύψει καθαρό ποσοστό.

Τέλος χρειάστηκε να ανακατασκευάσουμε το RDD σε DataFrame ορίζοντας το τελικό schema του ώστε να μπορέσει να γίνει αποθήκευσή του.