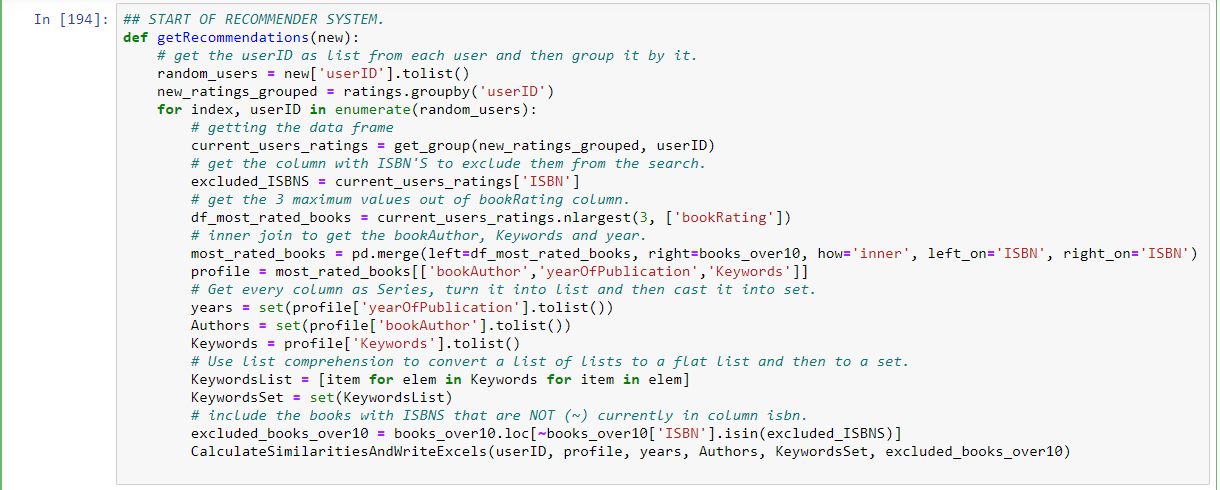
Μαρίνος Μάριος (Dai17147) Recommender Systems in Python.

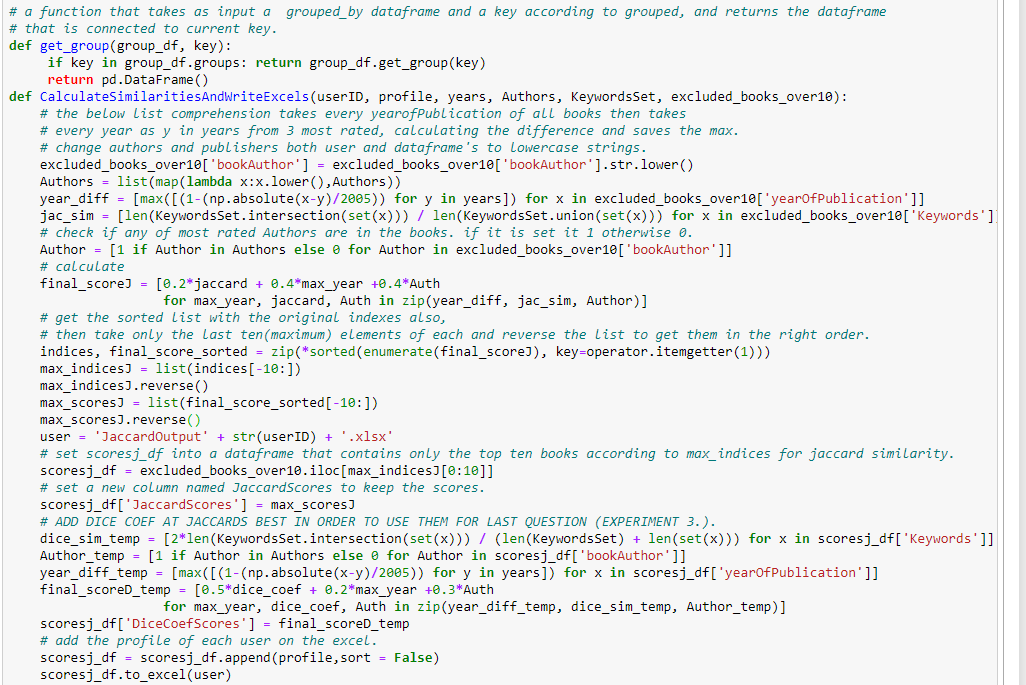
# Σχολιασμός Κώδικα.

Όσον αφορά τον κώδικα, χρησιμοποιήσα Python και την βιβλιοθήκη pandas συγκεκριμένα. Πέρα από την προεπεξεργασία που αναφέρεται, έκανα και κάποια πράματα ακόμη. Αρχικά, αφαίρεσα τις στήλες 'imageUrlS', 'imageUrlM', 'imageUrlL' μιας και δεν χρειάζονται πουθενά. Έπειτα, κοίταξα αν οι βαθμολογίες που υπάρχουν στο books\_ratings csv υπάρχουν και τα αντίστοιχα τους βιβλία στο books.csv οπότε όποιες βαθμολογίες δεν έχουν το αντίστοχο βιβλίο τις αφαίρεσα από το ratings dataframe. Το ίδιο έκανα και για τους χρήστες αλλά εκεί όλοι αντιστοιχούσαν. Έπειτα σε δύο συγκεκριμένα βιβλία είχε year = 'DK Publishing Inc' ως λάθος και βλέποντας τις αντίστοιχες εγγραφές παρατήρησα οτι ήταν όλα μία στήλη πιο δεξιά (δηλαδή στην στήλη bookAuthor είχε την τιμή του bookTitle κ.ο.κ.) από όσο έπρεπε άρα διόρθσα τις αντίστοιχες γραμμές. Το ίδιο ίσχυε για μία γραμμή με year = Gallimard και έκανα τα ίδια. Συνεχίζοντας, διόρθωσα τις τιμές των years που υπάρχουν και είναι άκυρες, δηλαδή τις τιμές πάνω από το 2005 μιας και το dataset δημιουργήθηκε τότε και το θεωρούμε ως άνω όριο ενώ και τα βιβλιά που έχουν year = 0 και τα αντικατέστησα με τον μέσο όλων των υπόλοιπων years. Έπειτα κοίταξα για ποιές γραμμές δεν έχουν τιμή στο bookAuthor και βρήκα οτι ήταν μία μόνο οπότε έψαξα ποιός είναι ο συγγραφέας και το αντικατέστησα. Για την εξαγωγή keywords από τα bookTitles χρησιμοποιήσα την βιβλιοθήκη rake\_nltk και πιο συγκεκριμένα το Rake. Η υπόλοιπη διαδικασία έχει ως εξής: έχω δύο συναρτήσεις η μία getReccomendations η οποία παίρνει ως είσοδο τους (n=5) τυχαίους χρήστες και για αυτούς τους 5 (για κάθε έναν ξεχωριστά) βρίσκει τα 3 καλύτερα βιβλία τουκαι δημιουργεί το προφιλ.



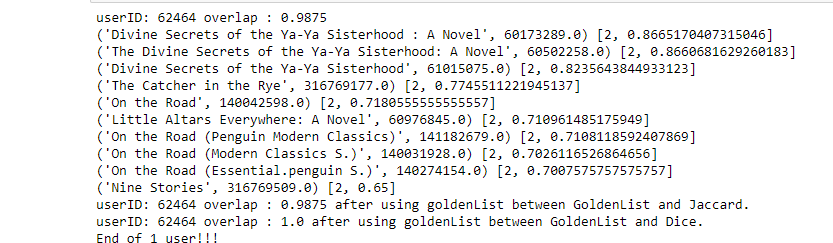
Εικόνα 1: Συνάρτηση GetReccomendations

Έπειτα καλείται για καθέ έναν ξεχωριστά μία συνάρτηση CalculateSimilaritiesAndWriteExcels η οποία κάνει αυτό που λέει. Υπολογίζει για κάθε χρήστη την ομοιότητα του για κάθε βιβλίο, επιλέγει τα 10 καλύτερα και τα γράφει σε ένα excel. Επίσης, για κάθε χρήστη **αφού** βρει τα 10 καλύτερά του βιβλία για αυτά τα 10 πάει και υπολογίζει **και** την Jaccard/Dice αντίστοιχα και το βάζει στο excel για να την χρησιμοποιήσω αργότερα για το πειραματικό μέρος ώστε να βρίσκω πιο ‘άνετα’ τον μέσο όρο μεταξύ των Dice + Jaccard για να το έχω για το 3ο ερώτημα στο πειραματικό μέρος σαν το 2ο κομμάτι ταξινόμησης μετά το πλήθος εμφανίσεών του κάθε βιβλίου. Παρακάτω στην εικόνα φαίνεται μόνο ο υπολογισμός για το Jaccard στα καλύτερα 10, και έπειτα **για αυτά** τα 10 κάνω και τον υπολογισμό του Dice coeff. Προφανώς παρακάτω στον κώδικα έχω και για το Dice/Jaccard.



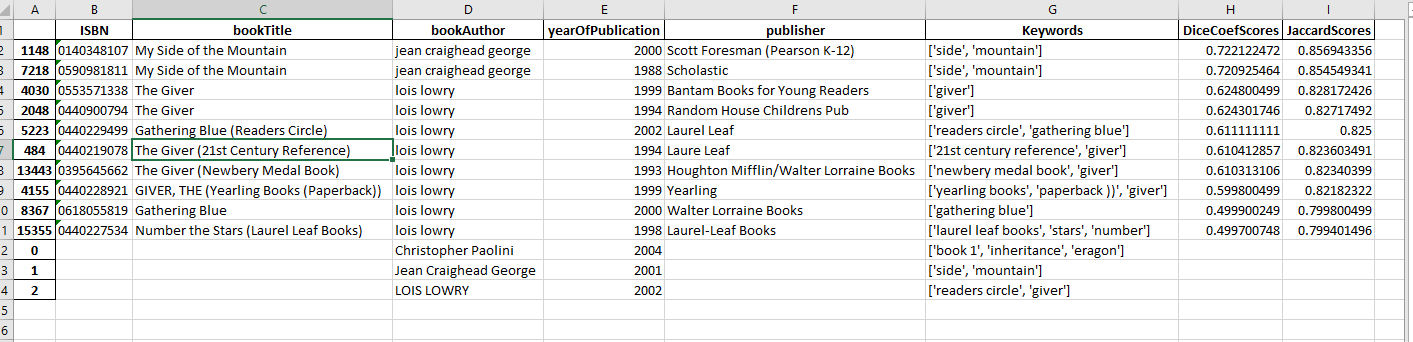
Εικόνα 2: Συνάρτηση CalculateSimilaritiesAndWriteExcels.

Τέλος, εδώ έπιλέγονται οι 5 τυχαίοι( σε σχόλιο αφήνω τους τυχαίους που μου έβγαλε για να το τρέξετε κ εσείς αν θέλετε. Ένα output **για έναν χρήστη από τους 5** είναι το παρακάτω το οποιό δείχνει πρώτα το overlap μεταξύ των δύο λιστών του, έπειτα την **χρυσή λίστα του** και από κάτω τα αντίστοιχα overlap μεταξύ της χρυσής λίστας και των δύο λιστών (Dice, Jaccard).



Εικόνα 3: Ένα τελικό output από έναν τυχαίο χρήστη

Άκομή ένα αρχείο excel(έστω για το DiceCoef) για τον χρήστη έχει την εξής δομή : Όπου έχει τα 10 βιβλία ταξινομημένα από το καλύτερο στο χειρότερο με βάσει το DiceCoefScores και δίπλα ακριβώς για αυτά τα 10 και την Jaccard ώστε να υπολογίσω μετά ευκολότερα τον μέσο όρο του. Στις τελευταίες 3 γραμμές υπάρχει **και μόνο του** το προφιλ του χρήστη.



Εικόνα 4: πάραδειγμα αρχείο εξόδου Excel για έναν τυχαίο χρήστη.

# Σχολιασμός και άλλες προσεγγίσεις.

Αρχικά, για τους 5 χρήστες υπάρχει μία τάση στο να υπάρχουν υψηλότερες βαθμολογίες χρησιμοποιηώντας την συνάρτηση με την Jaccard similarity σε αντίθεση με την DiceCoef όπου στην DiceCoef έχουμε βιβλίο με **32%** ομοιότητα μόνο που είναι πολύ λίγο(στον χρήστη 86001), το οποίο είναι και αναμενόμενο μιας και για παράδειγμα με την Jaccard μπορούμε να έχουμε **80% ομοιότητα** σε δύο βιβλία χωρίς να περιέχεται **καμία** λέξη κλειδί των βιβλίων του χρήστη με κάποιο από τα βιβλία ( αν το έτος και ο συγγραφέας είναι ακριβώς ίδιοι.), είναι προφανές οτι δίνεται πολύ βαρύτητα στο έτος και στον συγγραφέα ενώ από την άλλη στην συνάρτηση με το DiceCoef δίνεται πολύ βαρύτητα στις λέξεις-κλειδιά και για αυτό το λόγο έχουμε αρκετά μικρές βαθμολογίες μιας και φυσικό είναι να μην ανήκουν **όλες** οι λέξεις κλειδιά ενός βιβλίου στις λέξεις κλειδιά των 3 ‘καλύτερων’ βιβλιών του χρήστη. Φαίνεται άρα οτι οι δύο συναρτήσεις βασίζονται πάρα πολύ στον συγγραφέα και στο έτος οπότε θα μπορούσαν να γίνουν δίαφορες τροποποιήσεις ώστε να ισσοροπηθεί η εξίσωση για παράδειγμα βάζοντας και τον εκδοτικό οίκο(publisher) του κάθε βιβλίου όπου είναι η ίδια προσέγγιση με τον συγγραφέα (δηλ. αν είναι ο ίδιος εκδοτικός οίκος 1 αλλιώς 0), με **ίσους συντελεστές βαρύτητας 25%** στο καθένα (Author, publisher, Keywords, year) ώστε να έχουν όλα ισάξια βαρύτητα με διαφορετικούς τρόπους υπολογισμού ομοιότητας μεταξύ λέξεων-κλειδιών. Επίσης θα μπορούσαν να γίνουν πολλές ακόμα παραλλαγές βάζοντας μέσα και σαν παράμετρο την ηλικία των χρηστών ή ακόμα και βάσει το location κάθε χρήστη. Επίσης, θα μπορούσαμε να ‘παίξουμε’ με τους συντελεστές κάθε παραμέτρου και να δοκιμάσουμε διάφορα μέχρι να πάρουμε ένα ικανοποιητικό αποτέλεσμα σε περίπτωση που δεν μας αρκεί αυτό (ή αν έχουμε κάποιο threshold ώστε να δεχτούμε και τα δέκα βιβλία για παράδειγμα να μην δεχόμαστε βιβλία που δεν έχουν 60%-70% **συνολική** **ομοιότητα** ή λιγότερο.). Τέλος σε όλα τα παραπάνω σενάρια θα μπορούσαμε να αλλάξουμε τον τρόπο υπολογισμού της ομοιότητας των λέξεων-κλειδιών με κάποια άλλη μετρική ομοιότητας και όχι την Jaccard ή την DiceCoef όπως την ομοιότητα cosine,pearson,spearman και άλλες ώστε να δούμε ποιά δίνει τα βέλτιστα αποτελέσματα για τα συγκεκριμένα δεδομένα.