ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΛΗΡΟΦΟΡΙΚΗΣ ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Συστήματα Διαχείρισης Βάσεων Δεδομένων

Υπεύθυνος Καθηγητής: Ιωάννης Θεοδωρίδης

Βοηθοί: Ιωάννης Κοντούλης, Γεώργιος Θεοδωρόπουλος

Φοιτητές: Άγγελος Μάης (Π15077) – Παναγιώτης Μανιός (Π17065) – Νικόλαος Καραμποϊκης (Π17040)

Υλοποίηση του Task 1.4

**Task 1.4:**

**Outer Joins**

**Index-Nested-Loop Joins**

**Sort-Merge Join**

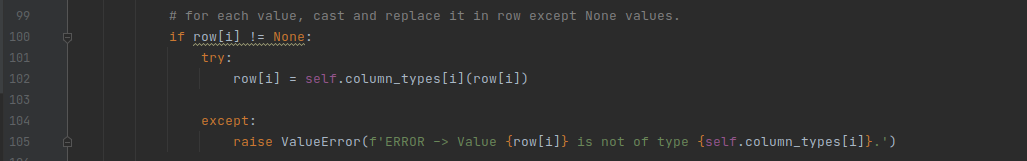
**Όλες οι λειτουργικές συναρτήσεις προστέθηκαν στην table.py και τα wrapper τους στο database.py.**

**Για την υλοποίηση του Outer Joins, δημιουργήσαμε τις 3 λειτουργίες του: Left Outer Join, Right Outer Join και Full Outer Join.**

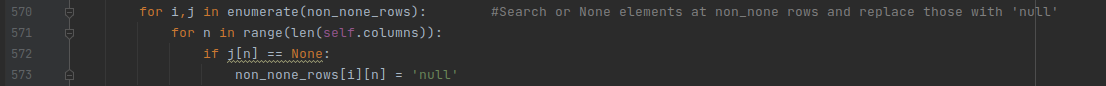
**Όσον αφορά το Left Outer Join αρχικά ανατρέξαμε τα στοιχεία της αριστερής λίστας και για κάθε της στοιχείο ψάχναμε την δεξιά λίστα. Όπου βρίσκαμε κοινά στοιχεία τα κάναμε Insert στην καινούρια λίστα μας (συγχώνευση των δύο λιστών λιστών), όμως άμα δεν υπήρχε κάποιο στοιχείο κοινό στοιχείο εισήγαμε την τιμή none σε μία καινούρια λίστα με μέγεθος όσο ήταν το πλήθος των στηλών της δεξιάς λίστας και ύστερα κάναμε insert τα none στοιχεία.**



**Τα none στοιχεία περνάνε τον έλεγχο της function \_insert με ένα if και είναι η μοναδική περίπτωση που περνάμε αυτόν τον έλεγχο.**



**Θελήσαμε όμως να εμφανίζονται στον χρήστη τα κενά στοιχεία ως null. Αυτό μπορέσαμε να το υλοποιήσουμε προσθέτοντας στην show ένα κομμάτι κώδικα που ψάχνει το None και αλλάζει σε null.**



**Η right\_outer\_join φτιάχτηκε ομοίως αλλά σαν λειτουργικότητα είχαμε βασική λίστα την δεύτερη και κάναμε τις ενέργειες από την δεύτερη την πρώτη.**

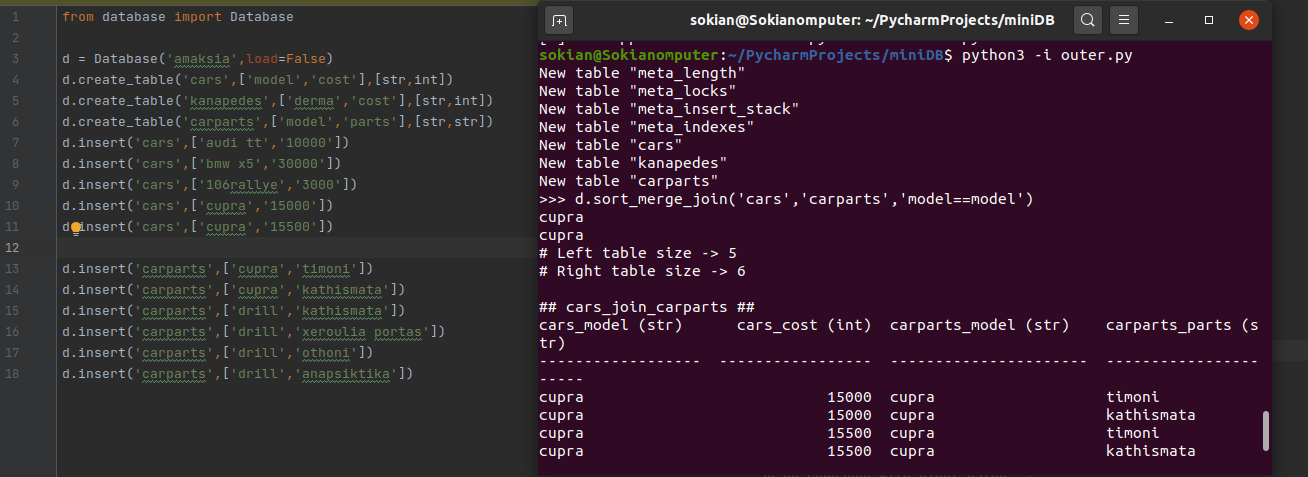
**Ενώ η full\_outer\_join είναι μια συγχώνευση των δύο functions**

**Το 3ο ερώτημα του Sort Merge Join δημιουργήσαμε μια function στην table.py, ύστερα κάναμε sort της δύο μας λίστες και φτιάξαμε ένα καινούριο table το οποίο θα είναι το τελικό table της sort\_merge. Φτάνοντας στο αλγοριθμικό κομμάτι δημιουργούμε δύο μεταβλητές που αναπαριστούν τα indexes σε κάθε μία από τις δύο λίστες, ύστερα συγκρίνουμε τα στοιχεία που παίρνουμε από τις λίστες και στην 1ο περίπτωση όπου και τα δύο στοιχεία είναι ίσα κάνουμε insert τις γραμμές με δείκτη i,j και ελέγχουμε αν είμαστε στο τέλος της δεύτερης λίστας, αν είμαστε κάνουμε break από το while αλλιώς αυξάνεται ο δείκτης j. Στην 2η περίπτωση όπου η τιμή του left\_table είναι μεγαλύτερη της τιμής του right\_table ελέγχουμε πάλι αν είμαστε στο τέλος της δεύτερης λίστας, αν είμαστε το while κάνει break, αλλιώς αυξάνει τον δείκτη j κατά 1. Στην 3η περίπτωση, αν το left\_value είναι μικρότερο του right\_value αυξάνουμε τον δείκτη i κατά 1 και συνεχίζουμε με ένα if για να ελέγξουμε αν το i είναι η τελευταία τιμή που μπορεί να πάρει αλλιώς θα υπερβεί το πλήθος της right list, αν περάσει τον έλεγχο θα γίνει ένα while με συνθήκη αν το j είναι μεγαλύτερο της μονάδας αφού θέλουμε να ελέγξουμε αν οι προηγούμενες τιμές της δεξιάς λίστας είναι μεγαλύτερες ή και ίσες της τιμής του left table, κι έτσι να μειώσουμε τον δείκτη j όσο χρειάζεται για να γίνουν οι απαραίτητες συγχωνεύσεις.**

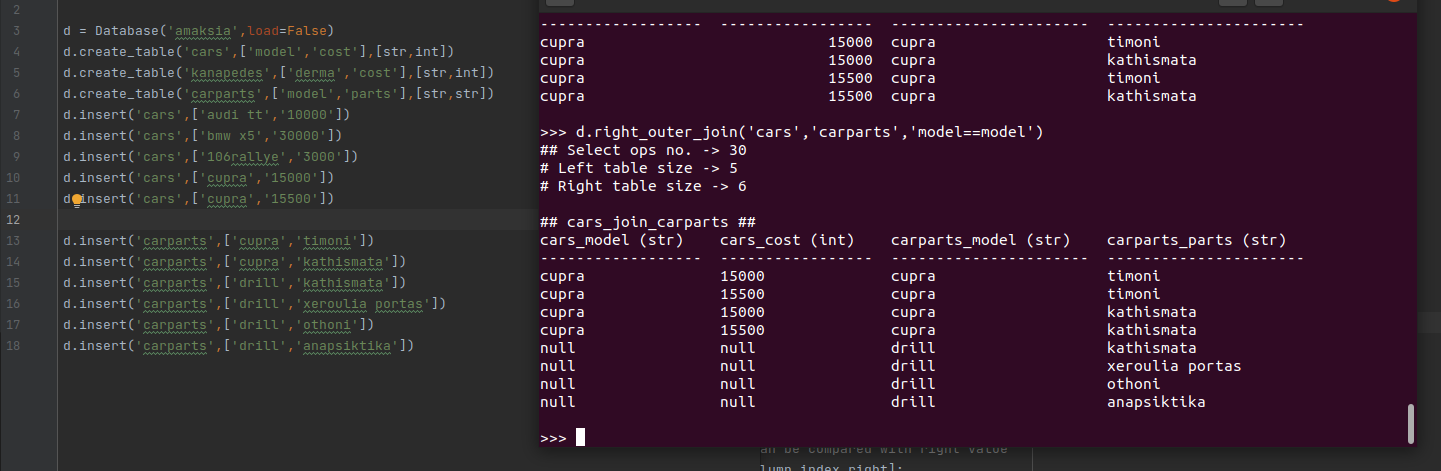


**Ενδεικτικά Screenshots με τις λειτουργίες που δημιουργήσαμε:**

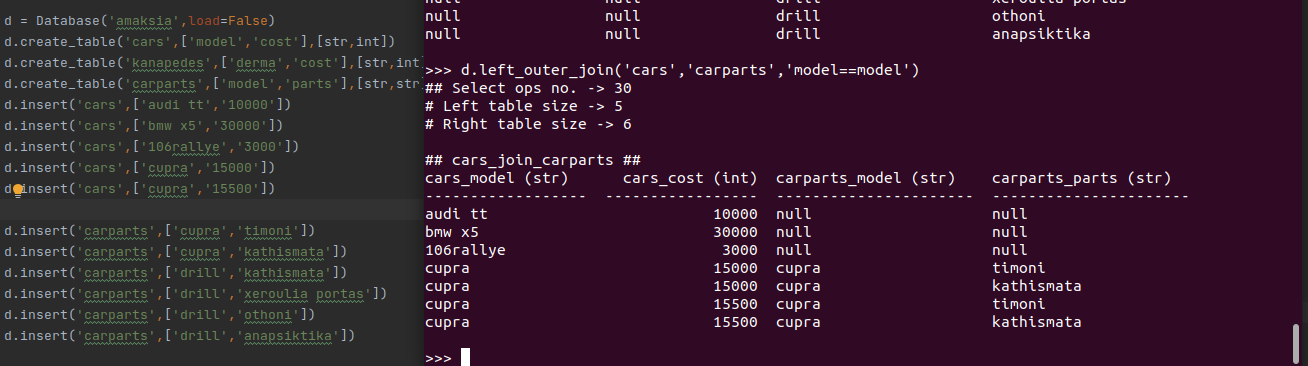
**Sort Merge Join**



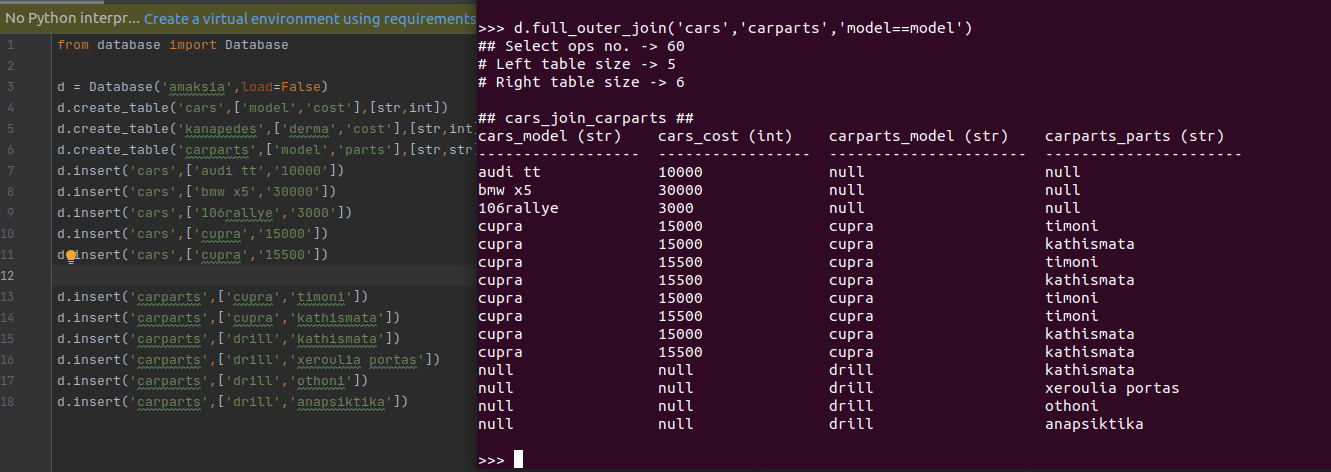
**Right Outer Join**



**Left Outer Join**



**Full Outer Join**



**Η λειτουργία του index nested loop join δεν υλοποιήθηκε καθόλου αφού μετά από αρκετή προσπάθεια δεν μπορέσαμε να καταλήξουμε σε κάποια ολοκληρωμένη λύση.**