**ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ**

**ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ**

**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**



**ΣΥΣΤΗΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ**

**5Ο ΕΞΑΜΗΝΟ**

**ΥΠΟΛΟΓΙΣΤΙΚΗ ΕΡΓΑΣΙΑ ΓΙΑ ΤΟ ΕΤΟΣ 2022-2023**

**ΟΜΑΔΑ 3 ΑΤΟΜΩΝ**

|  |  |  |
| --- | --- | --- |
| ΜΕΛΗ | ΟΝΟΜΑΤΕΠΩΝΥΜΟ | ΑΜ |
| 1ο ΜΕΛΟΣ | ΑΝΤΩΝΙΑ ΑΡΒΑΝΙΤΑΚΗ | Π19016 |
| 2ο ΜΕΛΟΣ | ΕΥΑΓΓΕΛΙΑ-ΑΦΡΟΔΙΤΗ ΜΠΑΡΔΟΠΟΥΛΟΥ | Π19110 |
| 3ο ΜΕΛΟΣ | ΚΛΕΟΠΑΤΡΑ ΤΣΟΥΜΕΑ | Π19177 |
|  |  |  |
|  |  |  |

1. **Issue 1**

Στα συγκεκριμένα ερωτήματα κληθήκαμε να εμπλουτίσουμε τη δήλωση WHERE υποστηρίζοντας διάφορους operators, οι οποίοι θα αναλυθούν παρακάτω. Για την ομαλή εκτέλεση των ερωτημάτων χρειάστηκε να προσθέσουμε επιπλέον λειτουργίες στα αρχεία mdb.py, database.py, misc.py και table.py.

**Misc.py:** Η συνάρτηση **def get\_op()** χρησιμοποιείται για να λάβουμε το op ως συνάρτηση των a,b χρησιμοποιώντας σύμβολο. Τα σύμβολα τα οποία προσθέσαμε εμείς είναι: μεγαλύτερο (>), not (!=), between και BETWEEN.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Ομοίως προσθέσαμε τα παραπάνω σύμβολα και στην συνάρτηση **def split\_condition()**. Στη συνέχεια, γίνεται έλεγχος αν το op\_key αποτελείται από τα σύμβολα που περιέχει και εκτελεί τις κατάλληλες λειτουργίες. Διαφορετικά, ελέγχει αν το op\_key είναι between, τότε κάνει split() στο condition και ελέγχει εάν υπάρχει το and στο query (εάν έχει γραφτεί σωστά) ώστε να πάρει όλες τις τιμές.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Έπειτα, δημιουργήσαμε τη συνάρτηση def between(), η οποία ελέγχει και επιστρέφει αντίστοιχα αν όλα τα στοιχεία είναι αλφαριθμητικά(ή αριθμοί) και τις αντίστοιχες τιμές που βρίσκονται ενδιάμεσα του a και του b.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Τέλος, ομοίως με τις συναρτήσεις get\_op(), split\_condition(), προσθέσαμε τους operators.

Εικόνα που περιέχει κείμενο, οθόνη, στιγμιότυπο οθόνης, ηλεκτρονικές συσκευές

Περιγραφή που δημιουργήθηκε αυτόματα

**Database.py:** Στη συνάρτηση **def select(),** δημιουργήσαμε έλεγχο για το condition. Αν το condition περιέχει το and ή το or τότε κάνει split() το condition και το επιστρέφει στο condition\_column.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, οθόνη, μαύρο

Περιγραφή που δημιουργήθηκε αυτόματα

**1.1.1 Issue 1a**

* **NOT**

**Table.py:** Η συνάρτηση **def \_parse\_condition()** ελέγχει αν το condition περιέχει τον operator not καλεί την συνάρτηση split\_condition() και επιστρέφει την τιμή με τον καθορισμένο τύπο στήλης και επιστρέφει το όνομα της στήλης, τον τελεστή και τη μεταφερόμενη τιμή.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

**ΕΚΤΕΛΕΣΗ**

**Graphical user interface, text

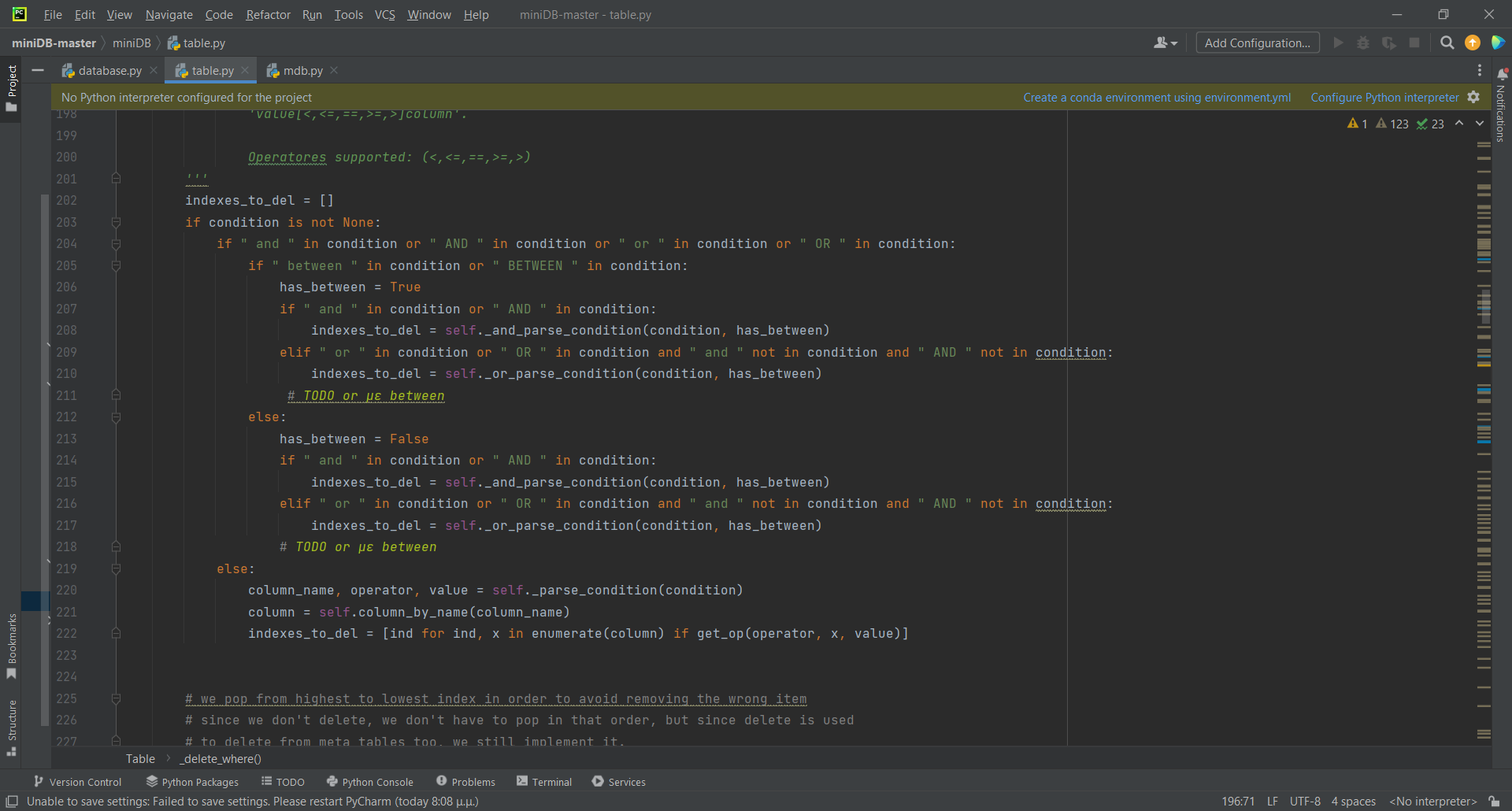
Description automatically generated**

**Text

Description automatically generated**

* **BETWEEN**

**Table.py:** Στη συνάρτηση **def \_delete\_where()** ελέγχουμε εάν το condition δεν είναι None και στη συνέχειαελέγχουμε επίσης εάν το condition περιέχει την τιμή and ή την τιμή or και επιπλέον την τιμή between. Αν ισχύει τότε καλεί την συνάρτηση **def \_and\_parse\_condition()** (για το and) ή την συνάρτηση **def \_or\_parse\_condition()** (για το or) και επιστρέφει τα στοιχεία στο indexes\_to\_del για να διαγραφτούν.

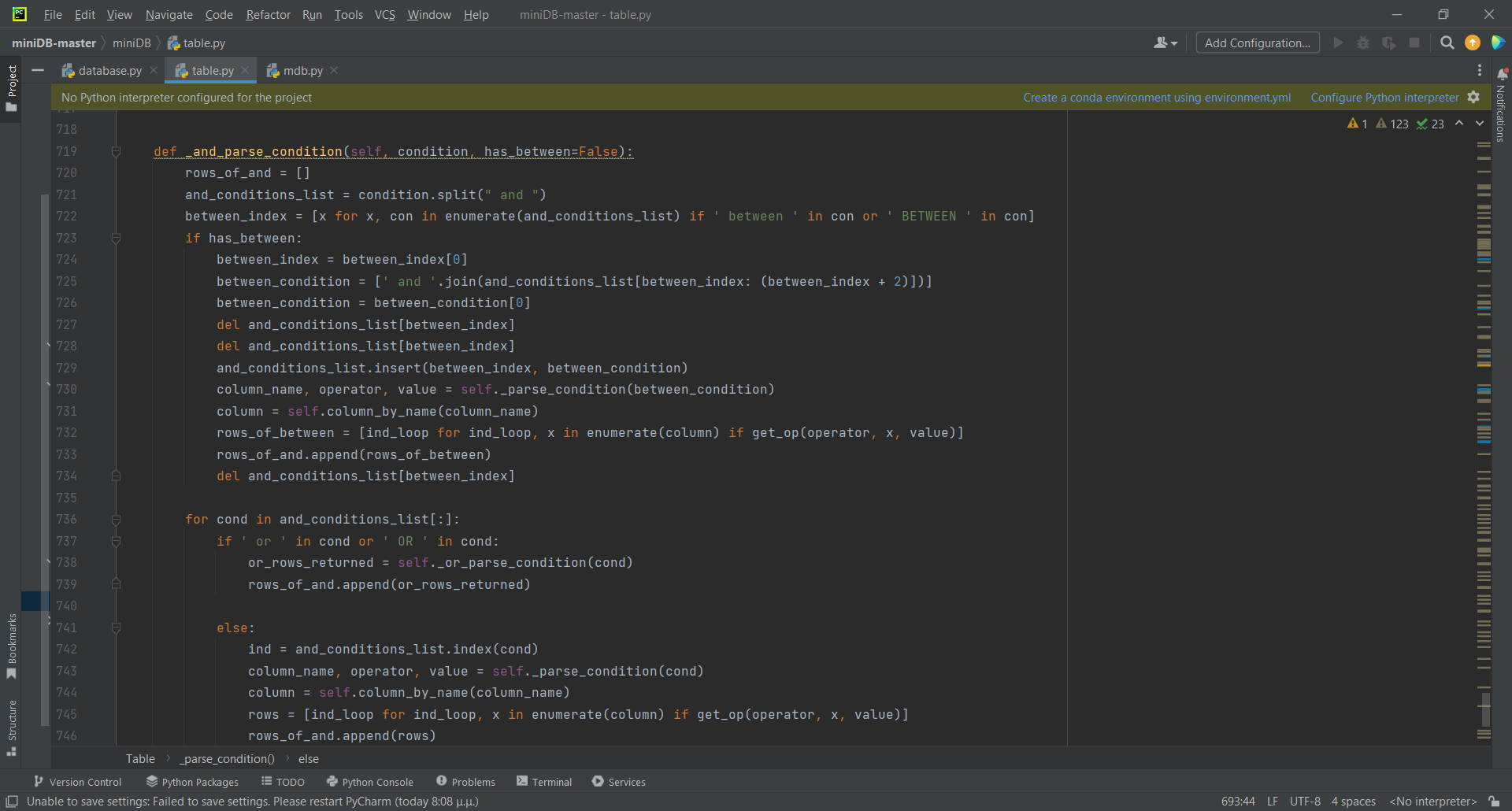


Ομοίως και στη συνάρτηση **def \_select\_where()** ελέγχουμε εάν το condition δεν είναι None και στη συνέχειαελέγχουμε επίσης εάν το condition περιέχει την τιμή and ή την τιμή or και επιπλέον την τιμή between. Αν ισχύει τότε καλεί τη συνάρτηση **def \_and\_parse\_condition()** (για το and) ή την συνάρτηση **def** **\_or\_parse\_condition()** (για το or) και επιστρέφει όλα τα στοιχεία που βρέθηκαν στη μεταβλητή row.

Εικόνα που περιέχει κείμενο, οθόνη, στιγμιότυπο οθόνης, μαύρο

Περιγραφή που δημιουργήθηκε αυτόματα

Η συνάρτηση **def \_and\_parse\_condition()** χρησιμοποιείται για να αναγνωρίζει τις συνθήκες που χωρίζονται με and στην περίπτωση όπου το condition περιέχει το between.

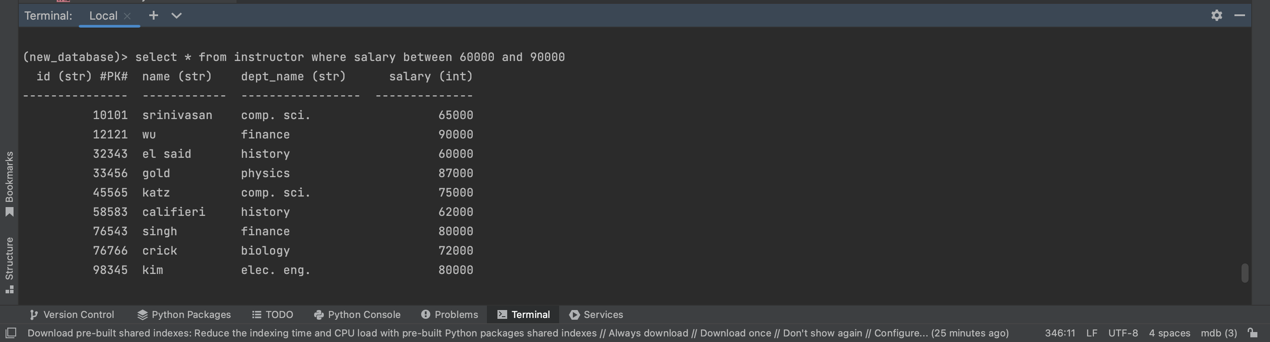


Η συνάρτηση **def \_or\_parse\_condition()** χρησιμοποιείται για να αναγνωρίζει τις συνθήκες που χωρίζονται με or στην περίπτωση όπου το condition περιέχει το between.

Εικόνα που περιέχει κείμενο

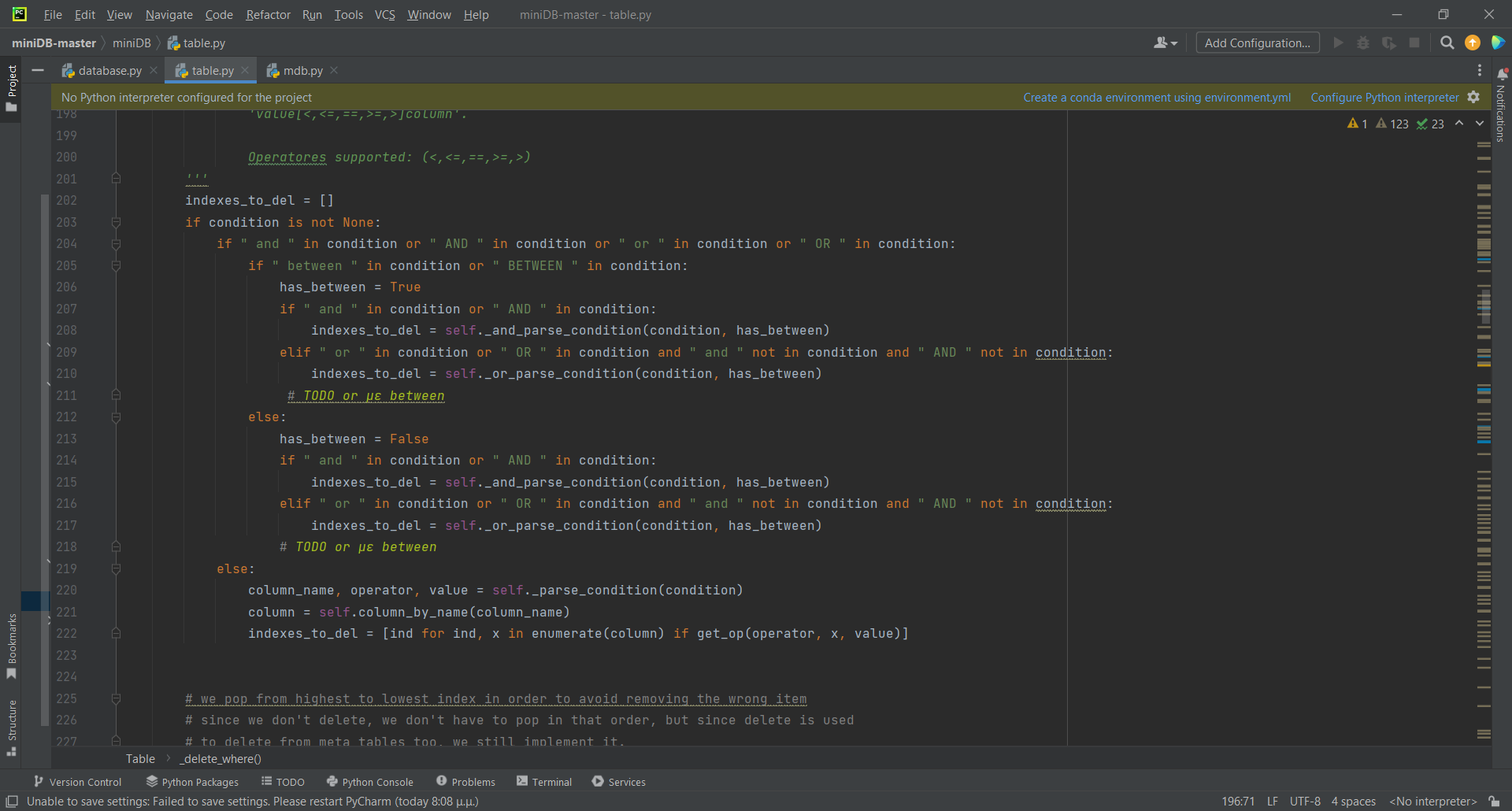
Περιγραφή που δημιουργήθηκε αυτόματα

**ΕΚΤΕΛΕΣΗ**

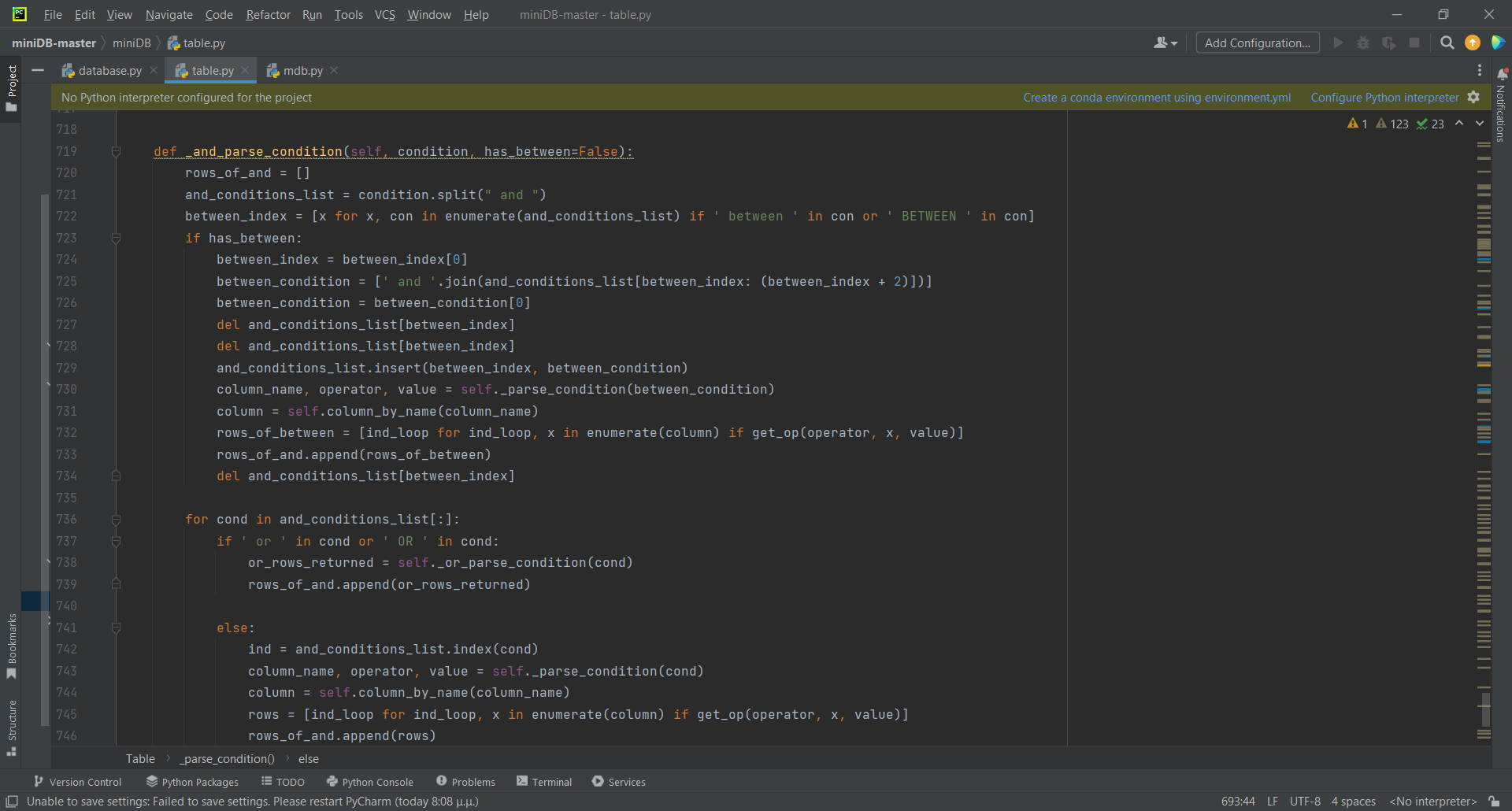
****

* **AND**

**Table.py:** Στη συνάρτηση **def \_delete\_where()** ελέγχουμε εάν το condition δεν είναι None και στη συνέχειαελέγχουμε επίσης εάν το condition περιέχει την τιμή and. Αν ισχύει τότε καλεί την συνάρτηση **def** **\_and\_parse\_condition**() και επιστρέφει τα στοιχεία στο indexes\_to\_del για να διαγραφτούν.



Για την υλοποίηση του συγκεκριμένου ερωτήματος δημιουργήσαμε την συνάρτηση **def \_and\_parse\_condition(),** όπου την χρησιμοποιούμε για να αναγνωρίζει τις συνθήκες που χωρίζονται με and.



Ομοίως και στη συνάρτηση **def \_select\_where()** ελέγχουμε εάν το condition δεν είναι None και στη συνέχειαελέγχουμε επίσης εάν το condition περιέχει την τιμή and. Αν ισχύει τότε καλεί τη συνάρτηση **def** **\_and\_parse\_condition()** και επιστρέφει όλα τα στοιχεία που βρέθηκαν στη μεταβλητή row.

Εικόνα που περιέχει κείμενο, οθόνη, στιγμιότυπο οθόνης, μαύρο

Περιγραφή που δημιουργήθηκε αυτόματα

**ΕΚΤΕΛΕΣΗ**

**Text

Description automatically generated**

* **OR**

**Table.py:** Στη συνάρτηση **def \_delete\_where()** ελέγχουμε εάν το condition δεν είναι None και στη συνέχειαελέγχουμε επίσης εάν το condition περιέχει την τιμή or. Αν ισχύει τότε καλεί την συνάρτηση **def** **\_or\_parse\_condition()** και επιστρέφει τα στοιχεία στο indexes\_to\_del για να διαγραφτούν.

Εικόνα που περιέχει κείμενο, οθόνη, στιγμιότυπο οθόνης, εσωτερικό

Περιγραφή που δημιουργήθηκε αυτόματα

Ομοίως και στη συνάρτηση **def \_select\_where()** ελέγχουμε εάν το condition δεν είναι None και στη συνέχειαελέγχουμε επίσης εάν το condition περιέχει την τιμή or. Αν ισχύει τότε καλεί τη συνάρτηση **def** **\_or\_parse\_condition()** και επιστρέφει όλα τα στοιχεία που βρέθηκαν στη μεταβλητή row.

Εικόνα που περιέχει κείμενο, οθόνη, στιγμιότυπο οθόνης, μαύρο

Περιγραφή που δημιουργήθηκε αυτόματα

Για την υλοποίηση του συγκεκριμένου ερωτήματος δημιουργήσαμε την συνάρτηση **def \_or\_parse\_condition(),** όπου την χρησιμοποιούμε για να αναγνωρίζει τις συνθήκες που χωρίζονται με or.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

**ΕΚΤΕΛΕΣΗ**

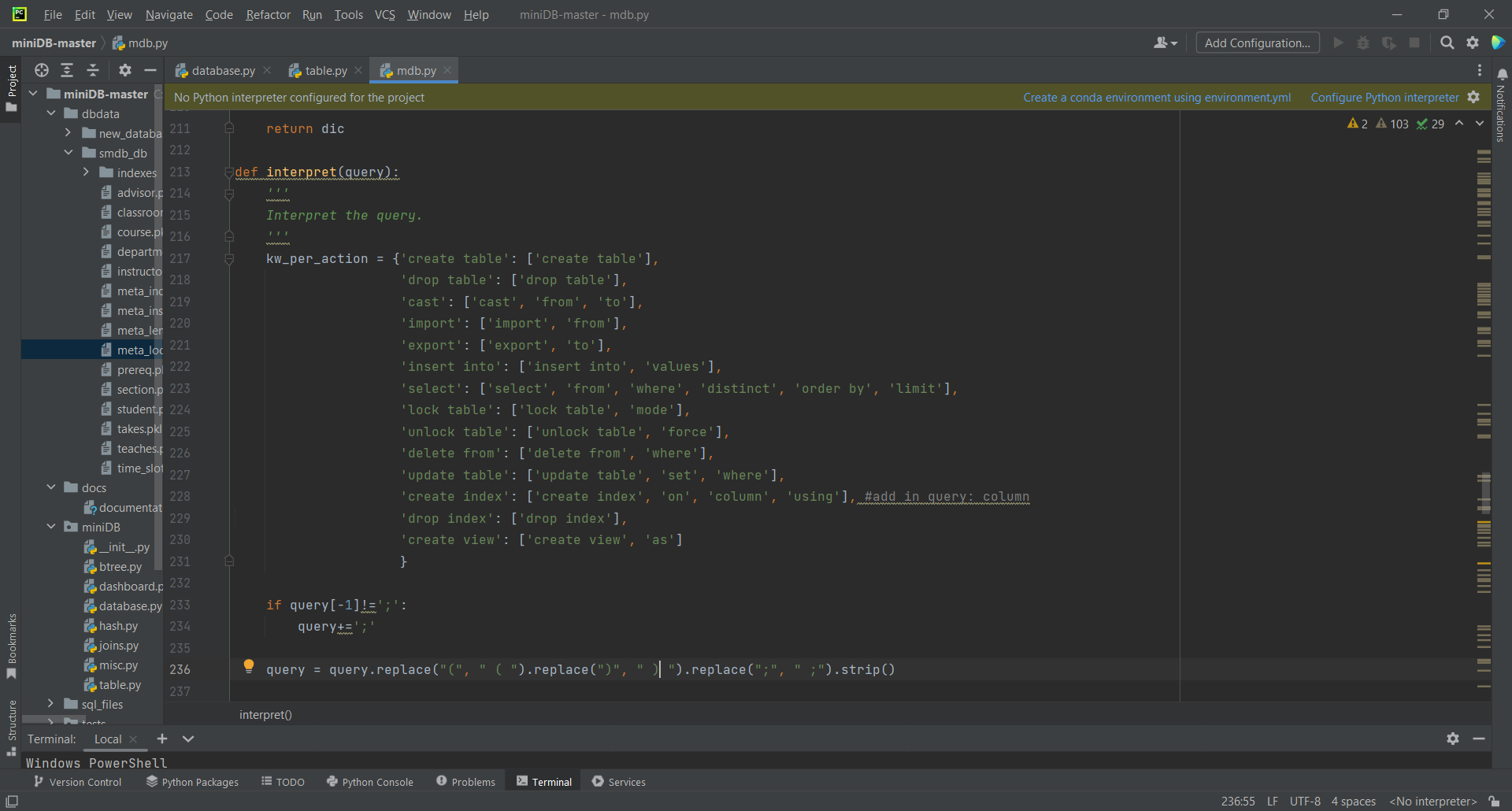
Text

Description automatically generated

1. **Issue 2**

Παρακάτω αναφέρονται οι προσθήκες που πραγματοποιήθηκαν για την εκτέλεση και τον δύο ερωτημάτων. Για την ομαλή εκτέλεση των ερωτημάτων χρειάστηκε να προσθέσουμε επιπλέον λειτουργίες στα αρχεία mdb.py, database.py και table.py. Ακόμη, έγινε μία προσθήκη στον πίνακα meta\_indexes (meta\_indexes.pkl). Παρακάτω αναλύονται όλες οι επιπλέον λειτουργίες των αρχείων που αναφέρθηκαν παραπάνω.

* **Mdb.py:** Στο συγκεκριμένο αρχείο και πιο συγκεκριμένα στηνσυνάρτηση **def interpret()**, προσθέσαμε μέσα στο **create index** (key\_per\_action) το column, ώστε κατά την εκτέλεση του query να μπορεί να υποστηρίξει την στήλη για την οποία γίνεται η εύρεση με btree ή με ευρετήριο κατακερματισμού.



Στον έλεγχο που περιέχει το create table, ελέγχουμε επιπλέον εάν το args περιέχει το unique τότε κάνει split() διαφορετικά επιστρέφει None.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Τέλος, προσθέσαμε ακόμη έναν επιπλέον έλεγχο εάν το action είναι create index, τότε προσθέτουμε μέσα στο dic[‘on’] to kw\_in\_query[1]. Στη συνέχεια εάν είναι διαφορετικό του κενού κάνει split() το dic[‘on’] και στη συνέχεια αφαιρεί τις παρενθέσεις.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

* **Database.py:** Έπειτα, στην main συνάρτηση της συγκεκριμένης κλάσης **(def \_\_init\_\_())** κατά τη δημιουργία του πίνακα **meta\_indexes** προσθέσαμε ακόμα μία επιπλέον στήλη, με όνομα **column\_name** που αποτελείται από αλφαριθμητικά. Η συγκεκριμένη στήλη χρησιμοποιείται για τις εμφανίζει τα indexes (unique στήλες) του πίνακα με τα οποία γίνεται η εύρεση, είτε btree είτε hash.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, υπολογιστής

Περιγραφή που δημιουργήθηκε αυτόματα

Στη συνάρτηση, **def create\_table()** προσθέσαμε επιπλέον την παράμετρο **unique\_columns=None**, καθώς και την κάναμε update ώστε να μπορεί να υποστηρίζεται.

Έπειτα, τροποποιήσαμε τη συνάρτηση **def select()**. Αρχικά, κατά τον παρακάτω έλεγχο προσθέσαμε επιπλέον αν το condition\_column αποτελείται από unique στήλη.

Εικόνα που περιέχει κείμενο, οθόνη, στιγμιότυπο οθόνης, εσωτερικό

Περιγραφή που δημιουργήθηκε αυτόματα

* **Table.py:** Αρχικά, στη συνάρτηση main (\_\_init\_\_) προσθέσαμε ως παράμετρο unique\_columns όπου την ορίζουμε ως None.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, οθόνη, ηλεκτρονικές συσκευές

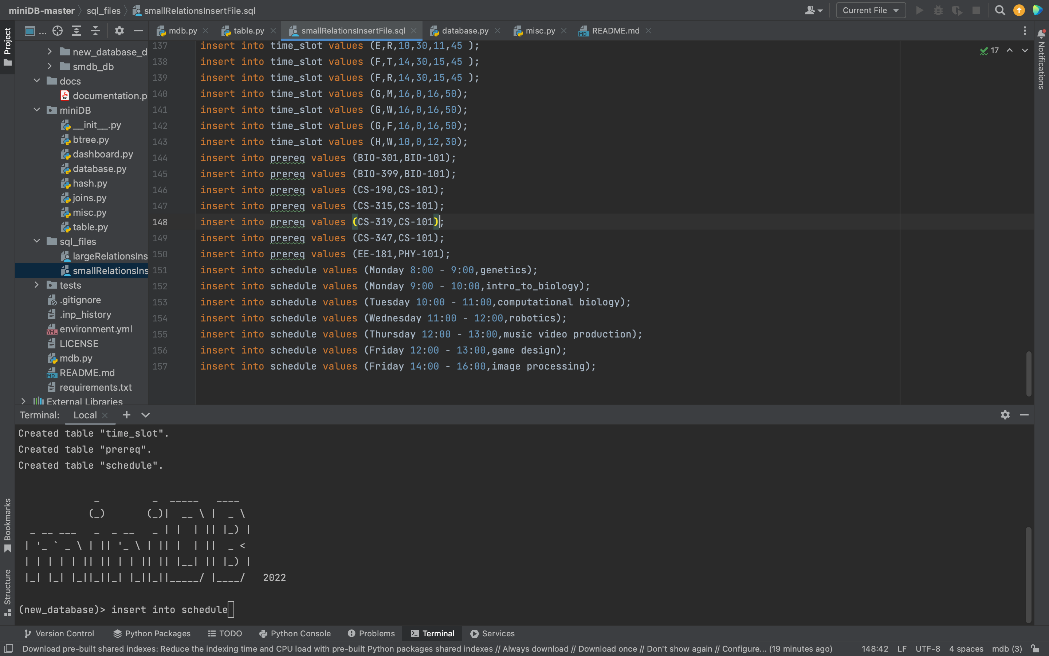
Περιγραφή που δημιουργήθηκε αυτόματα

Ομοίως, όπως πραγματοποιείται έλεγχος για το primary key έτσι και για το unique\_columns. Ελέγχουμε εάν έχει οριστεί και στη συνέχεια διατηρούμε το ευρετήριο του ως χαρακτηριστικό.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, οθόνη, υπολογιστής

Περιγραφή που δημιουργήθηκε αυτόματα

Τέλος, στο sql αρχείο **smallRelationsInsertFile** προσθέσαμε πραγματοποιήθηκαν επιπλέον inserts για τις unique στήλες τις οποίες θα χρειαστούμε στο συγκεκριμένο ερώτημα.



**2.1.1 Issue 2a**

Στο συγκεκριμένο ερώτημα κληθήκαμε να εμπλουτίσουμε τη λειτουργικότητα ευρετηρίασης υποστηρίζοντας το BTree index πάνω σε unique στήλες (non-PK).

**Database.py:** Για τη συνάρτηση **def select()**, όπως αναφέρθηκε παραπάνω προσθέσαμε επιπλέον αν το condition\_column αποτελείται από unique στήλη. Στην περίπτωση που ισχύει, περάσαμε όλα τα στοιχεία του πίνακα meta\_indexes και στη συνέχεια με την βοήθεια της δομής επανάληψης for πήραμε τα τελικά στοιχεία (\_selecte\_where\_with\_btree()) που θέλουμε όταν βέβαια η εύρεση γίνεται με btree. Τέλος, αν δεν ισχύει καμία από τις παραπάνω περιπτώσεις εμφανίζεται exception και ενημερώνει τον χρήστη ότι ο πίνακας δεν περιέχει btree index.

Εικόνα που περιέχει κείμενο, οθόνη, στιγμιότυπο οθόνης, εσωτερικό

Περιγραφή που δημιουργήθηκε αυτόματα

Στη συνάρτηση **def create\_index()**, προσθέσαμε την παράμετρο column\_name για να αναγνωρίζεται η στήλη. Γίνεται έλεγχος αν η στήλη είναι unique και στη συνέχεια δημιουργεί ευρετήριο Btree, όταν το index\_type είναι btree.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Στη συνάρτηση **def \_construct\_index()**, όταν το index\_type είναι btree, εισάγουμε την τιμή και το ευρετήριο στο btree για κάθε εγγραφή στο πρωτεύον κλειδί του πίνακα. Διαφορετικά δημιουργούμε ευρετήριο στην unique στήλη και εισάγουμε την τιμή και το ευρετήριο της στο btree.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

**2.1.2 ΕΚΤΕΛΕΣΗ Issue 2a**

Δημιουργία btree index σε unique column

Graphical user interface, text, application

Description automatically generated

Αδυναμία δημιουργίας btree index σε στήλη που δεν είναι unique ή primary key

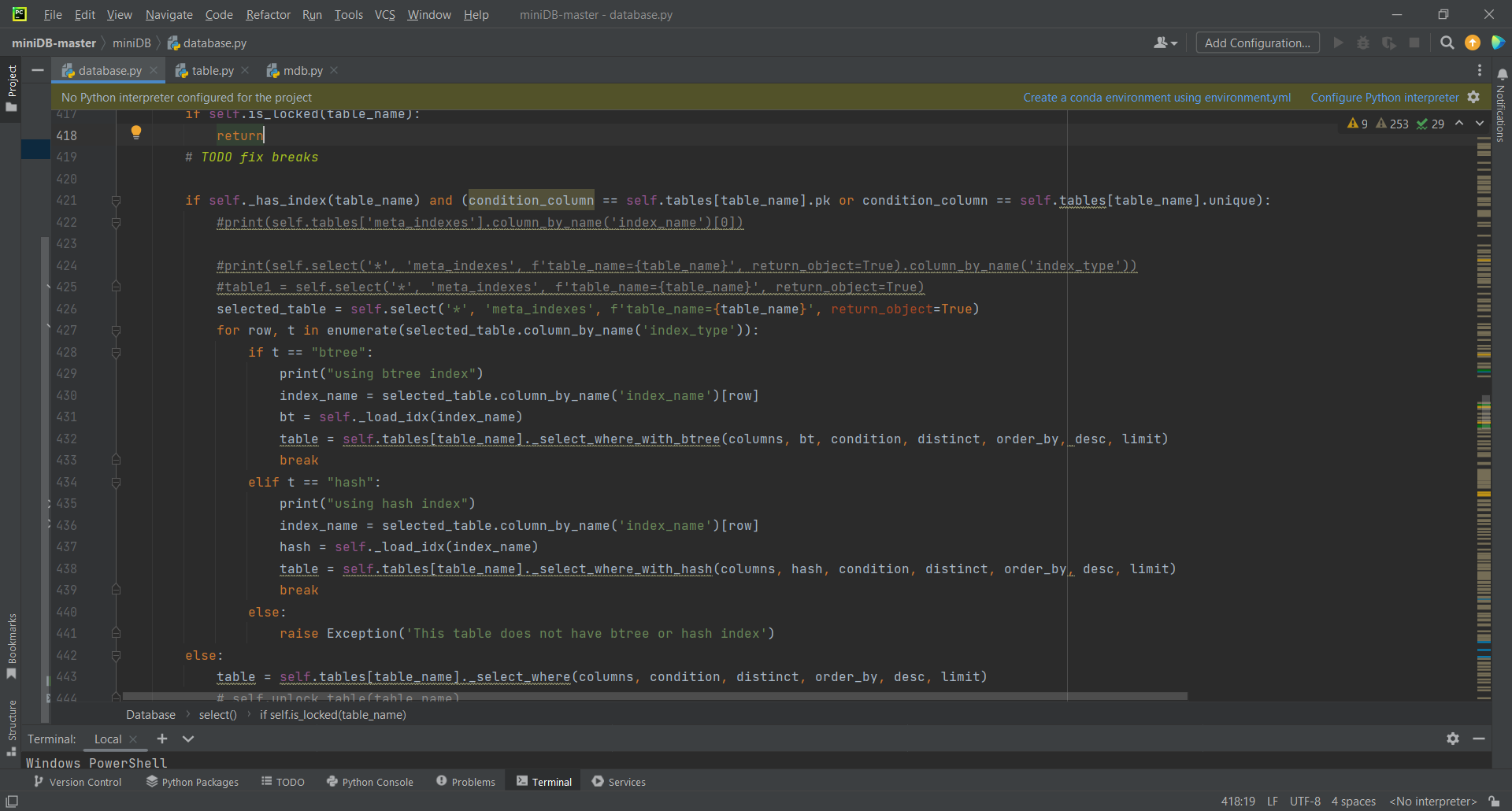
Text

Description automatically generated

**2.2.1 Issue 2b**

Στο συγκεκριμένο ερώτημα κληθήκαμε να εμπλουτίσουμε τη λειτουργικότητα ευρετηρίασης υποστηρίζοντας το ευρετήριο κατακερματισμού σε unique στήλες (non-PK).

**Database.py:** Για τη συνάρτηση **def select()**, όπως αναφέρθηκε παραπάνω προσθέσαμε επιπλέον αν το condition\_column αποτελείται από unique στήλη. Στην περίπτωση που ισχύει, περάσαμε όλα τα στοιχεία του πίνακα meta\_indexes και στη συνέχεια με την βοήθεια της δομής επανάληψης for, πήραμε τα τελικά στοιχεία (\_selecte\_where\_with\_hash()) που θέλουμε όταν βέβαια η εύρεση γίνεται με hash. Τέλος, αν δεν ισχύει καμία από τις παραπάνω περιπτώσεις εμφανίζεται exception και ενημερώνει τον χρήστη ότι ο πίνακας δεν περιέχει btree index.



Στη συνάρτηση **def create\_index()**, προσθέσαμε την παράμετρο column\_name για να αναγνωρίζεται η στήλη. Γίνεται έλεγχος αν η στήλη είναι unique και στη συνέχεια δημιουργεί ευρετήριο hash, όταν το index\_type είναι hash.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Στη συνάρτηση **def \_construct\_index()**, όταν το index\_type είναι hash, εισάγουμε την τιμή και το ευρετήριο στο hash για κάθε εγγραφή στο πρωτεύον κλειδί του πίνακα. Διαφορετικά δημιουργούμε ευρετήριο στην unique στήλη και εισάγουμε την τιμή και το ευρετήριο της στο hash.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, οθόνη, υπολογιστής

Περιγραφή που δημιουργήθηκε αυτόματα

**Table.py:** Δημιουργήσαμε τη συνάρτηση **def \_select\_where\_with\_hash()**, η οποία στην περίπτωση που το return\_column είναι \* επιστρέφει όλες τις στήλες. Διαφορετικά, βρίσκουμε τα ευρετήρια των στηλών για τις καθορισμένες στήλες. Έπειτα, ελέγχουμε εάν πρόκειται για ένα ερώτημα εύρους που δεν υποστηρίζεται από hash και πραγματοποιεί διαδοχική αναζήτηση. Διαφορετικά, εάν είναι point query βρίσκει το ευρετήριο της γραμμής της στήλης που ισούται με την τιμή.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

**2.2.2 ΕΚΤΕΛΕΣΗ Issue 2b**

Δημιουργία btree index σε unique column

Text

Description automatically generated

Αδυναμία δημιουργίας btree index σε στήλη που δεν είναι unique ή primary key

Text

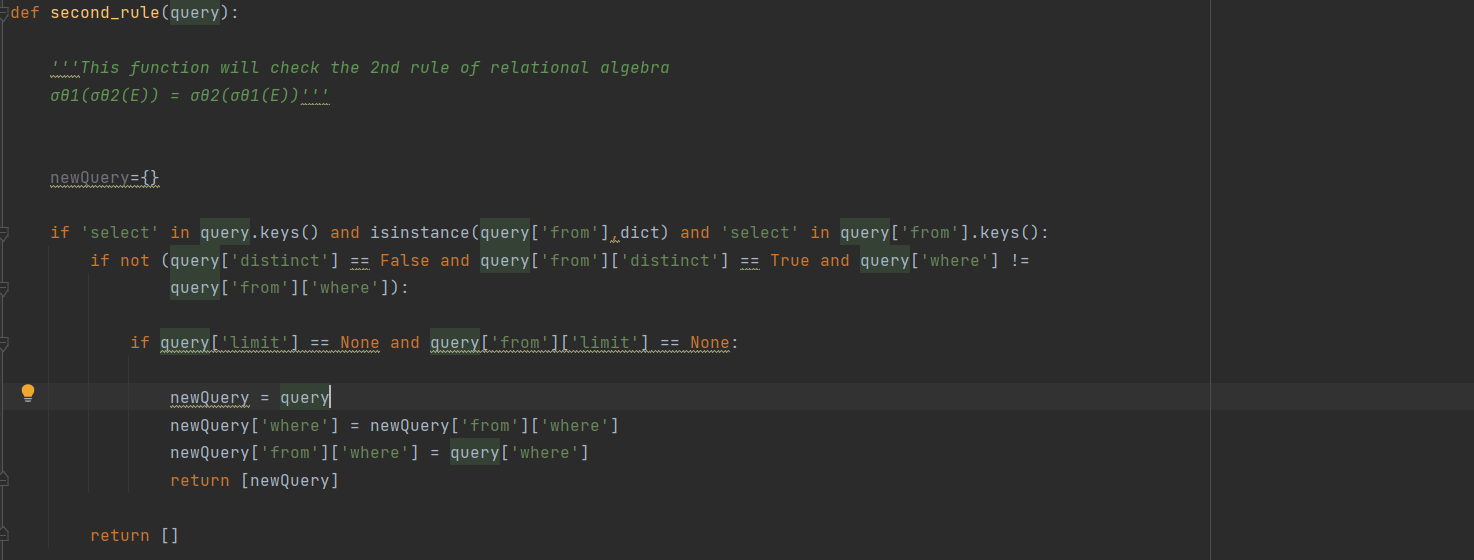
Description automatically generated

1. **Issue 3**

**3.1 Issue 3a**

Η συνάρτηση **second\_rule(query)** που βρίσκεται στο αρχείο algebra\_rules.py, αφορά τον δεύτερο κανόνα της σχεσιακής άλγεβρας, του αρχείου equivalenceRules.dvi και συγκεκριμένα τον κανόνα, σθ1 (σθ2 (E)) = σθ2 (σθ1 (E)).

Η second\_rule, δέχεται ένα query ως παράμετρο και σκοπός της είναι να ελέγξει εάν αυτός ο κανόνας εφαρμόζεται. Όταν ο κανόνας δεν μπορεί να εφαρμοστεί μας επιστρέφεται μια κενή λίστα η newQuery, η οποία στην περίπτωση που εφαρμόζεται ο κανόνας επιστρέφεται με τα δεδομένα που προστέθηκαν από το query που δόθηκε αρχικά.



Η συνάρτηση **fifth\_rule(query)** που βρίσκεται στο αρχείο algebra\_rules.py, αφορά τον δεύτερο κανόνα της σχεσιακής άλγεβρας, του αρχείου equivalenceRules.dvi και συγκεκριμένα τον κανόνα, E1 ⊲⊳θ E2 = E2 ⊲⊳θ E.

Η συνάρτηση αυτή, παίρνει ως όρισμα ένα query, και το αντιγράφει σε μια νέα μεταβλητή την newQuery. Εάν η λέξη ‘join’ βρίσκεται μέσα στο query που μας δόθηκε ως παράμετρος, τότε ενεργοποιείται ένα νέο εμφωλευμένο if, που ελέγχει εάν το join που μας δόθηκε είναι left ή right. Εάν είναι right τότε το newQuery παίρνει το left join, εάν είναι left τότε παίρνει το right. Αντίστοιχα, μετά αυτή η αλλαγή γίνεται και για το newQuery και στο τέλος η συνάρτηση μας επιστρέφει το newQuery, που είναι η νέα αντιστοίχιση που δημιουργήθηκε.

