



ARISTOTLE
UNIVERSITY
OF THESSALONIKI



SCHOOL OF
INFORMATICS

**Data & Web
Science**



MSc Program

ADVANCED TOPICS IN DATABASES

Εργασία στις Χωρικές Βάσεις Δεδομένων
Εαρινό Εξάμηνο 2018 – 2019

ΛΙΑΠΙΚΟΣ ΘΕΟΔΩΡΟΣ, ΑΜ: 11

ΠΕΡΙΕΧΟΜΕΝΑ

1. <u>ΕΙΣΑΓΩΓΗ</u>	2
1.1 <u>Περιβάλλον Ανάπτυξης</u>	2
1.2 <u>Δεδομένα (χωρικά και μη) που χρησιμοποιήθηκαν</u>	2
2. <u>ΠΕΡΙΓΡΑΦΗ ΒΑΣΗΣ ΧΩΡΙΚΩΝ ΔΕΔΟΜΕΝΩΝ</u>	3
2.1 <u>Περιγραφή των Χωρικών Δεδομένων</u>	3
2.2 <u>Περιγραφή των μη Χωρικών Δεδομένων</u>	5
3. <u>ΔΙΑΓΡΑΜΜΑ ΟΝΤΟΤΗΤΩΝ – ΣΥΣΧΕΤΙΣΕΩΝ</u>	7
4. <u>ΣΧΕΣΙΑΚΟ ΜΟΝΤΕΛΟ ΔΕΔΟΜΕΝΩΝ</u>	8
5. <u>ΜΗ ΧΩΡΙΚΑ ΕΡΩΤΗΜΑΤΑ</u>	9
6. <u>ΧΩΡΙΚΑ ΕΡΩΤΗΜΑΤΑ</u>	16
7. <u>ΠΑΡΑΡΤΗΜΑ</u>	26
8. <u>ΒΙΒΛΙΟΓΡΑΦΙΑ – ΑΝΑΦΟΡΕΣ</u>	33

1. ΕΙΣΑΓΩΓΗ

Στην παρούσα εργασία επιλέχθηκε να μελετηθεί μια Βάση Δεδομένων (ΒΔ), η οποία έχει αποθηκευμένα χωρικά δεδομένα. Η ΒΔ αποκτήθηκε έτοιμη και συμπληρωμένη από σχετικό διαδικτυακό τόπο και έγινε αφαίρεση των στοιχείων-πινάκων που ήταν χρήσιμα για την εργασία. Η ΒΔ εμπλουτίστηκε επιπλέον με συνθετικά μη χωρικά δεδομένα, που κρίθηκαν απαραίτητα για την σημασιολογική πληρότητα, αλλά και για την εκτέλεση των σχετικών μη χωρικών ερωτημάτων.

1.1 Περιβάλλον Ανάπτυξης

Η εργασία ολοκληρώθηκε σε περιβάλλον Λειτουργικού Συστήματος Linux Mint 19.0, με Desktop Environment Cinnamon 3.8.9. Η αποθήκευση των δεδομένων έγινε σε PostgreSQL 2.5.2, ενώ την υποστήριξη χωρικών δεδομένων την παρείχε το PostGIS 2.5.2. Ως πλατφόρμα διαχείρισης και ανάπτυξης της ΒΔ, αλλά και εκτέλεσης των ερωτημάτων, χρησιμοποιήθηκε το pgAdmin 4.8-2. Η οπτικοποίηση των χωρικών δεδομένων έγινε σε QGIS 2.18.17.

Για τη σχεδίαση του Διαγράμματος Οντοτήτων – Συσχετίσεων αλλά και του Σχεσιακού Μοντέλου Δεδομένων χρησιμοποιήθηκε το σχεδιαστικό πρόγραμμα yEd 3.19.

1.2 Δεδομένα (χωρικά και μη) που χρησιμοποιήθηκαν

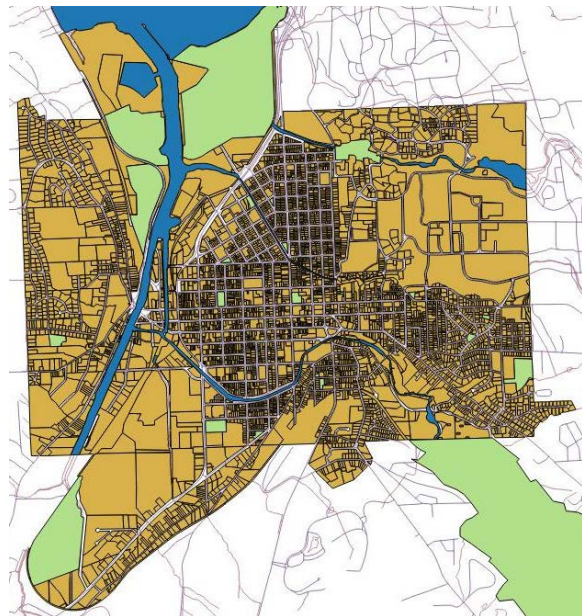
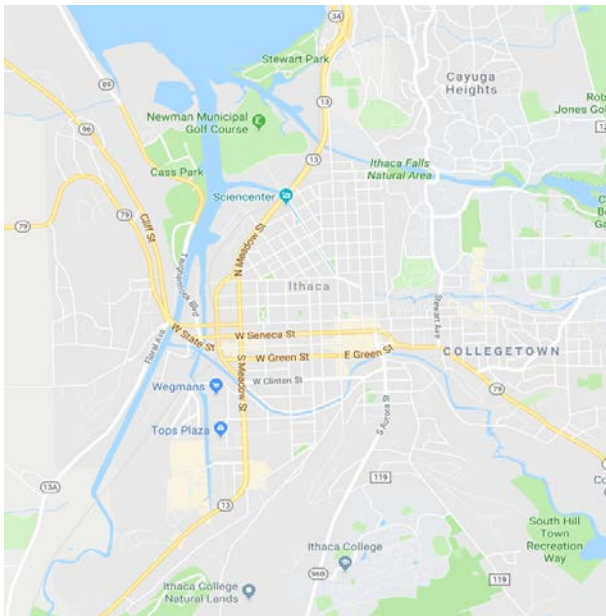
Για την εργασία χρησιμοποιήθηκε έτοιμη ΒΔ χωρικών δεδομένων, η οποία περιγράφεται αναλυτικά στην επόμενη ενότητα. Πριν τη χρήση της, η ΒΔ τροποποιήθηκε, ώστε να περιέχει τα απολύτως απαραίτητα στοιχεία:

- Αφαιρέθηκαν οι πίνακες που δεν θα χρειαστούν. Ο κώδικας SQL της συγκεκριμένης διαδικασίας εμφανίζεται στο Παράρτημα, καθώς και στο συνοδευτικό αρχείο *'Drop unnecessary tables.sql'*.
- Δημιουργήθηκαν νέοι πίνακες για τα μη χωρικά δεδομένα, αλλά και για να ικανοποιηθούν οι απαιτήσεις του Σχεσιακού Μοντέλου Δεδομένων. Ο κώδικας SQL της συγκεκριμένης διαδικασίας εμφανίζεται στο Παράρτημα, καθώς και στο συνοδευτικό αρχείο *'Creation of necessary tables.sql'*.
- Δημιουργήθηκαν συνθετικά μη χωρικά δεδομένα για να συμπληρώσουν με τους μη χωρικούς πίνακες. Ο κώδικας Python της συγκεκριμένης διαδικασίας εμφανίζεται στο Παράρτημα, καθώς και στο συνοδευτικό αρχείο *'Populate tables.py'*.
- Τα συνθετικά δεδομένα αποθηκεύτηκαν σε αρχεία .csv και έγινε η εισαγωγή τους στους κατάλληλους πίνακες μέσω του pgAdmin.
- Έγινε εξαγωγή της τελικής δομής της ΒΔ μέσω του pgAdmin, ώστε να μπορέσει να χρησιμοποιηθεί από τον Καθηγητή για τον έλεγχο της λειτουργικότητας. Το αρχείο που περιέχει το backup ονομάζεται *'Spatial_DB.backup'* και βρίσκεται στο φάκελο 'Συνοδευτικά αρχεία'.

Τα ερωτήματα, που αφορούσαν τόσο τα χωρικά όσο και τα μη χωρικά δεδομένα, εκτελέστηκαν μέσα από το pgAdmin. Ο κώδικας SQL της συγκεκριμένης διαδικασίας εμφανίζεται στο Παράρτημα, καθώς και στα συνοδευτικά αρχεία *'Non Spatial Queries.sql'* και *'Spatial Queries.sql'*.

2. ΠΕΡΙΓΡΑΦΗ ΒΑΣΗΣ ΧΩΡΙΚΩΝ ΔΕΔΟΜΕΝΩΝ

Η ΒΔ που χρησιμοποιήθηκε περιέχει πλήθος δεδομένων, χωρικά και μη χωρικά, για το οικισμό Ithaca της πολιτείας της Νέας Υόρκης των Η.Π.Α. Παρακάτω φαίνεται η συγκεκριμένη περιοχή τόσο στο Google Maps, αλλά και όπως οπτικοποιείται από τα δεδομένα της ΒΔ στο QGIS.



2.1 Περιγραφή των Χωρικών Δεδομένων

Ακολουθούν οι κυριότερες χωρικές οντότητες (entities) που περιλαμβάνονται στη ΒΔ, καθώς και τα πεδία που περιέχουν τα σχετικά με αυτές δεδομένα. Κάποιες από αυτές δεν θα χρησιμοποιηθούν στον περιορισμένο αριθμό ερωτημάτων (queries) που θα εκτελεστούν, αλλά αναφέρονται χάριν πληρότητας.

parcels: Αντιστοιχούν στα αγροτεμάχια που περιλαμβάνει η περιοχή. Κύρια πεδία:

oid: Το πρωτεύον κλειδί του πίνακα.

acres: Η έκταση του αγροτεμαχίου σε στρέμματα.

asmt: Η εκτίμηση της συνολικής αξίας του αγροτεμαχίου (αξία γης και κτηρίων) σε δολάρια.

geometry: Τα χωρικά δεδομένα του αγροτεμαχίου.

parks: Αντιστοιχούν στα πάρκα που περιλαμβάνει η περιοχή. Κύρια πεδία:

name: Το πρωτεύον κλειδί του πίνακα.

geometry: Τα χωρικά δεδομένα του πάρκου.

waterway: Οι υδάτινες επιφάνειες (λίμνες και ποτάμια) που περιλαμβάνει η περιοχή. Κύρια πεδία:

oid: Το πρωτεύον κλειδί του πίνακα.

geometry: Τα χωρικά δεδομένα του υδάτινου πόρου.

floodarea: Οι περιοχές πλημμύρας που περιλαμβάνει η περιοχή. Κύρια πεδία:

oid: Το πρωτεύον κλειδί του πίνακα.

geometry: τα χωρικά δεδομένα της περιοχής πλημμύρας.

roads: Οι δρόμοι που περιλαμβάνει η περιοχή. Κύρια πεδία:

oid: Το πρωτεύον κλειδί του πίνακα.

geometry: Τα χωρικά δεδομένα του δρόμου.

trees: Τα δέντρα που περιλαμβάνει η περιοχή. Κύρια πεδία:

geometry: Τα χωρικά δεδομένα του δέντρου.

firm: Αντιστοιχεί στην ταξινόμηση της περιοχής σε ζώνες, ανάλογα με την επικινδυνότητα εμφάνισης πλημμύρας (lood insurance ranking map). Κύρια πεδία:

oid: Το πρωτεύον κλειδί του πίνακα.

zone: Οι ζώνες ταξινόμησης.

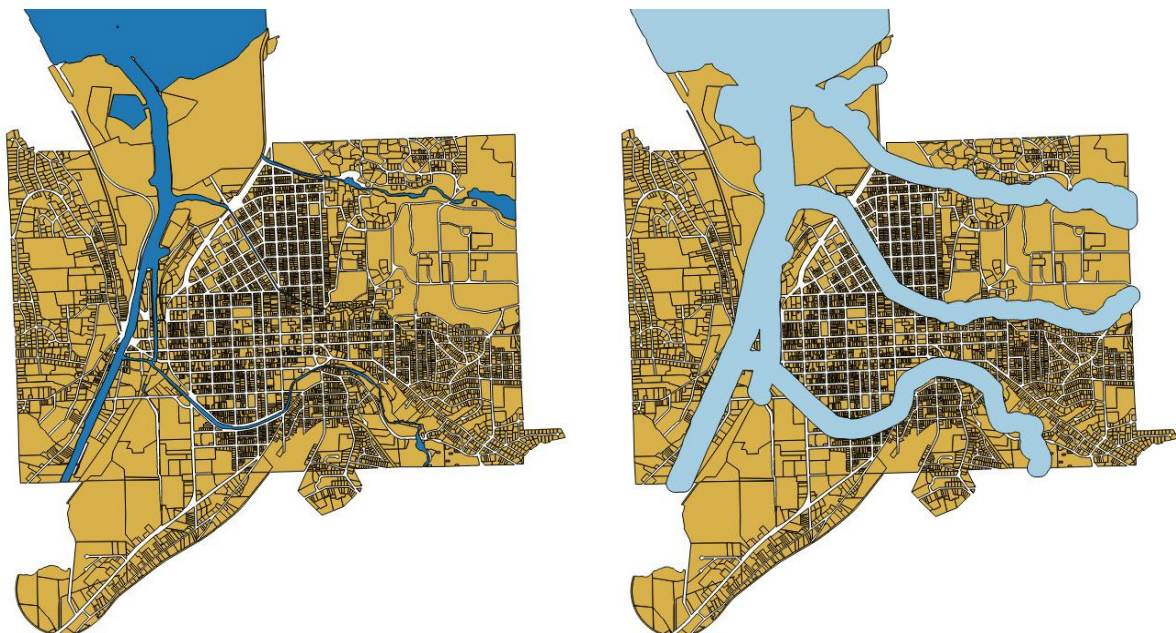
geometry: τα χωρικά δεδομένα των ζωνών.

Παρακάτω φαίνεται η οπτική αναπαράσταση των πιο χαρακτηριστικών παραπάνω χωρικών οντοτήτων από το QGIS

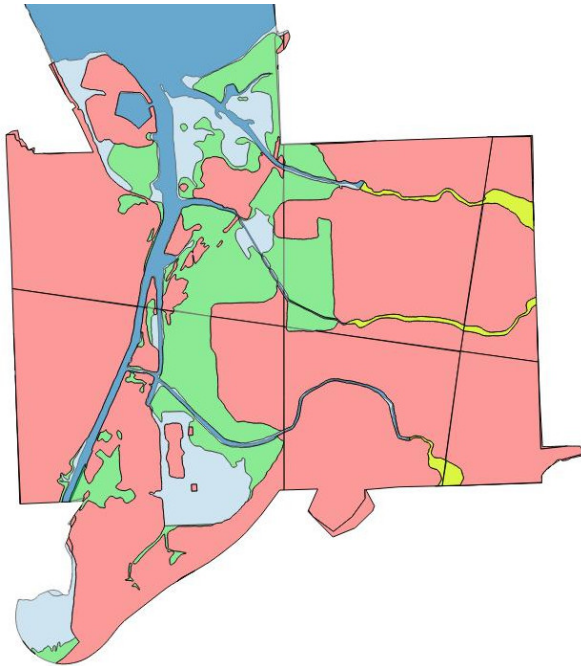
Τα αγροτεμάχια (parcels) αριστερά και τα πάρκα (parks) δεξιά:



Οι υδάτινες επιφάνειες (waterway) αριστερά και οι περιοχές πλημμύρας (floodarea) δεξιά:



Οι διάφορες ζώνες επικινδυνότητας εμφάνισης πλημμυρών (firm):



2.2 Περιγραφή των μη Χωρικών Δεδομένων

Παράλληλα με τα παραπάνω Χωρικά Δεδομένα δημιούργησα και μια σειρά συνθετικών μη Χωρικών Δεδομένων, τα οποία και πρόσθεσα στη συνολική ΒΔ.

people: Οι άνθρωποι που ζουν και εργάζονται στα αγροτεμάχια. Κύρια πεδία:

name: Το πρωτεύον κλειδί του πίνακα.

parcel_id: Εξωτερικό κλειδί από τον πίνακα parcels.

plants: Τα φυτά που καλλιεργούνται στα αγροτεμάχια. Κύρια πεδία:

name: Το πρωτεύον κλειδί του πίνακα.

season: η εποχή καλλιέργειας του κάθε φυτού.

annual_performances: Οι ετήσιες αποδόσεις των διαφόρων καλλιεργειών στα αγροτεμάχια. Κύρια πεδία:

ap_id: Το πρωτεύον κλειδί του πίνακα.

parcel_id: Εξωτερικό κλειδί από τον πίνακα parcels.

plant_id: Εξωτερικό κλειδί από τον πίνακα plants.

performance: Η ετήσια απόδοση μιας συγκεκριμένης καλλιέργειας σε ένα συγκεκριμένο αγροτεμάχιο, σε χιλιάδες δολάρια.

parc_cult_plants: Πίνακας που δημιουργείται από το Σχεσιακό Μοντέλο εξαιτίας του M:N τύπου συσχέτισης ανάμεσα στους τύπους Οντοτήτων Parcels και Plants του διαγράμματος Οντοτήτων-Συσχετίσεων. Κύρια πεδία:

parcel_id: Εξωτερικό κλειδί από τον πίνακα parcels.

plant_id: Εξωτερικό κλειδί από τον πίνακα plants.

Τα δύο εξωτερικά κλειδιά αποτελούν και το συνολικό πρωτεύον κλειδί του πίνακα.

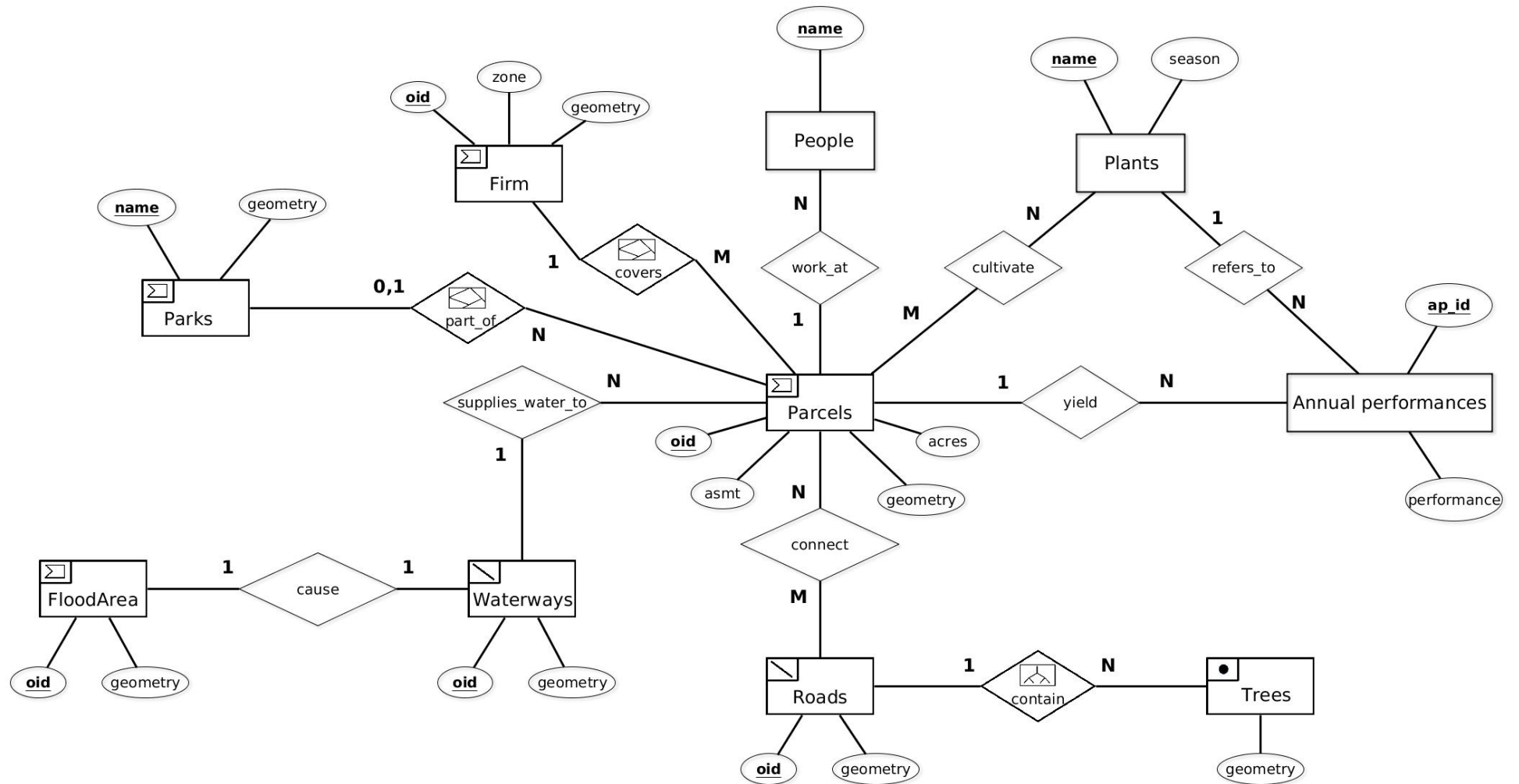
road_con_parc: Πίνακας που δημιουργείται από το Σχεσιακό Μοντέλο εξαιτίας του M:N τύπου συσχέτισης ανάμεσα στους τύπους Οντοτήτων Parcels και Roads του διαγράμματος Οντοτήτων-Συσχετίσεων. Κύρια πεδία:

parcel_id: Εξωτερικό κλειδί από τον πίνακα parcels.

road_id: Εξωτερικό κλειδί από τον πίνακα roads.

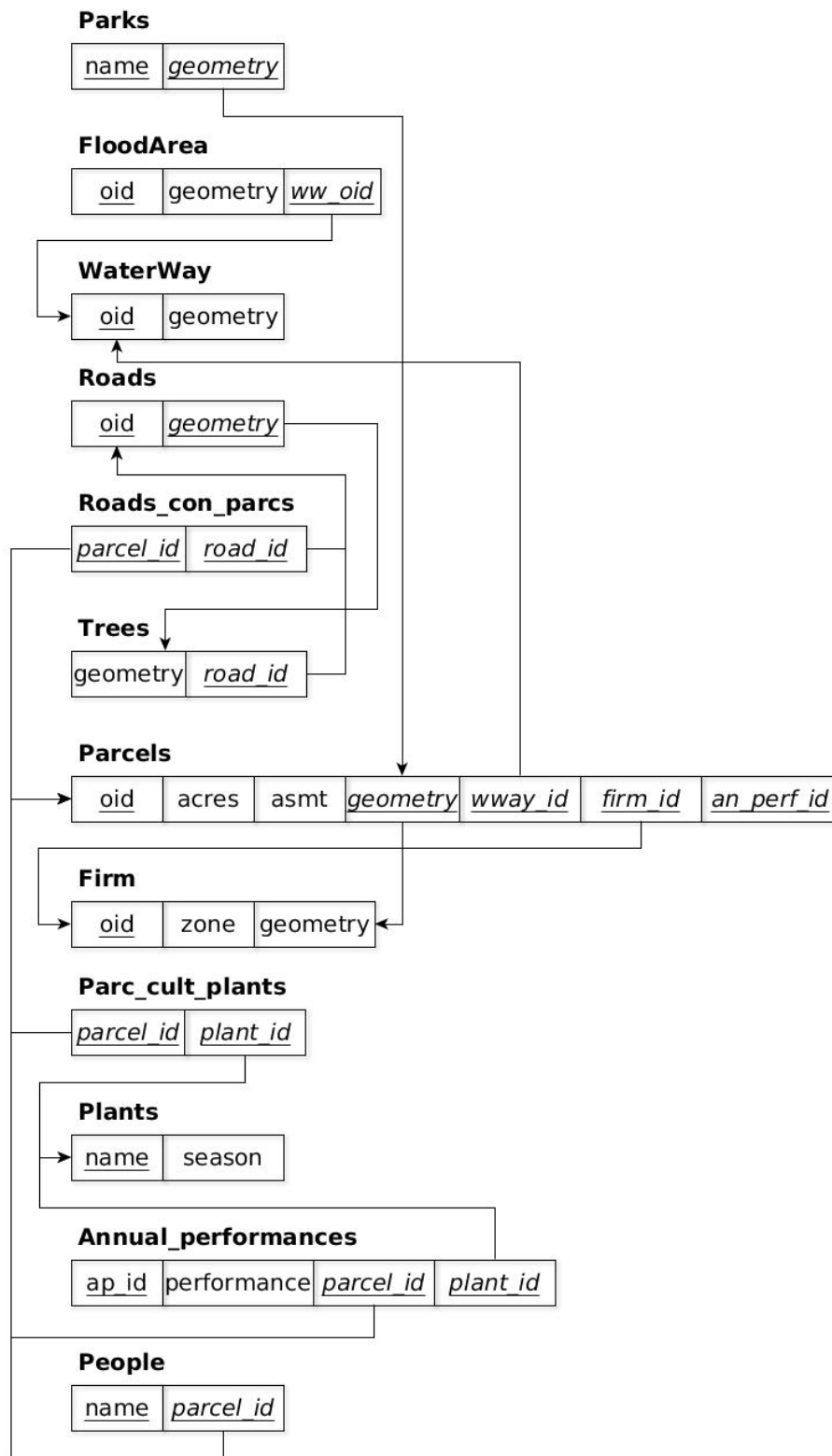
Τα δύο εξωτερικά κλειδιά αποτελούν και το συνολικό πρωτεύον κλειδί του πίνακα.

3. ΔΙΑΓΡΑΜΜΑ ΟΝΤΟΤΗΤΩΝ – ΣΥΣΧΕΤΙΣΕΩΝ



4. ΣΧΕΣΙΑΚΟ ΜΟΝΤΕΛΟ ΔΕΔΟΜΕΝΩΝ

Εφαρμόζονται οι κλασικοί κανόνες μετατροπής διαγράμματος Οντοτήτων – Συσχετίσεων σε Σχεσιακό Μοντέλο Δεδομένων. Λόγω περιορισμών στο σχεδιαστικό λογισμικό που χρησιμοποιήθηκε, τα ξένα κλειδιά εμφανίζονται με κανονική υπογράμμιση και *italics* και όχι με διακεκομμένη διαγράμμιση, όπως θα έπρεπε.



5. ΜΗ ΧΩΡΙΚΑ ΕΡΩΤΗΜΑΤΑ

Θα εκτελέσω σειρά από μη χωρικά ερωτήματα με χρήση των κατάλληλων τελεστών, όπως αυτοί που ζητούνται στην εκφώνηση. Στο τέλος κάθε ερωτήματος θα εμφανίζεται και η έξοδος της επιλογής explain του pgAdmin, για την εκτέλεση του συγκεκριμένου ερωτήματος.

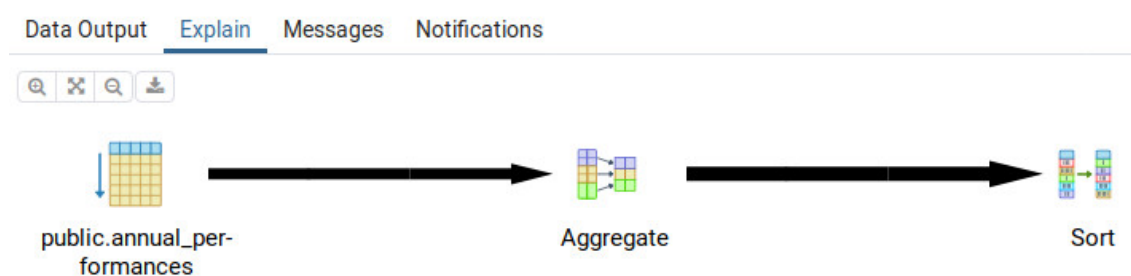
MX1. Χρήση του τελεστή συνάθροισης SUM() σε στοιχεία που έχουν ομαδοποιηθεί με τον τελεστή ομαδοποίησης GROUP_BY()

Ερώτημα: Να υπολογιστεί η συνολική απόδοση όλων των αγροκτημάτων σε όλα τα είδη καλλιέργειας. Τα αποτελέσματα να ταξινομηθούν σε φθίνουσα σειρά με βάση τη συνολική απόδοση.

```
SELECT parcel_id, SUM(performance) AS total_performance
FROM annual_performances
GROUP BY parcel_id
ORDER BY SUM(performance) DESC
```

Data Output	Explain	Messages	Notifications
	parcel_id integer	total_performance bigint	
1	151	2963	
2	1748	2917	
3	5524	2891	
4	1404	2837	
5	3614	2827	
6	5429	2810	
7	647	2803	
8	1203	2803	
9	2434	2802	
10	3343	2799	
11	1602	2799	
12	205	2797	
13	760	2795	
14	3181	2791	
15	2484	2790	
16	3145	2790	
17	1539	2776	

Έξοδος explain:



MX2. Χρήση του τελεστή συνάθροισης COUNT() σε στοιχεία που έχουν ομαδοποιηθεί με τον τελεστή ομαδοποίησης GROUP_BY(). Χρήση του τελεστή συνάθροισης MAX(). Χρήση εμφωλευμένου ερωτήματος.

Ερώτημα: Πόσοι είναι οι περισσότεροι άνθρωποι, που απασχολούνται σε κάποιο αγρόκτημα.

```
SELECT MAX(employees)
FROM(
    SELECT parcel_id, COUNT(*) AS employees
    FROM people
    GROUP BY parcel_id
    ORDER BY COUNT(*) DESC
) AS T
```

Το εμφωλευμένο ερώτημα επιστρέφει αρχικά λίστα με τα id των αγροκτημάτων και το σύνολο των ανθρώπων που απασχολούνται σε αυτά:

Data Output	Explain	Messages	Notifications
parcel_id integer	employees bigint		
1	2746	8	
2	511	8	
3	1120	7	
4	2630	7	
5	1007	7	
6	2943	7	
7	1458	7	
8	1895	7	
9	5050	7	

Τέλος, το εξωτερικό ερώτημα επιστρέφει το μεγαλύτερο νούμερο που εμφανίζεται στη στήλη employees:

Data Output	Explain	Messages	Notifications
max_employees bigint			
1	8		

Έξοδος explain:




MX3. Χρήση του τελεστή συνάθροισης MIN(). Χρήση εμφωλευμένου ερωτήματος.

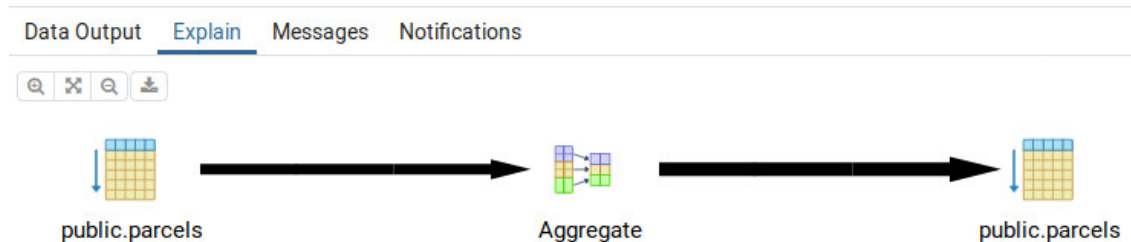
Ερώτημα: Ποιο είναι το αγρόκτημα με τη μικρότερη (μη μηδενική) αξία ανά στρέμμα.

```
SELECT parcel_id, asmt/acres AS dols_per_acre
FROM parcels
WHERE asmt/acres = (SELECT MIN(T.dols_per_acre)
                    FROM(SELECT asmt/acres AS dols_per_acre
                        FROM parcels
                        WHERE asmt/acres <> 0) AS T)
```

Χρησιμοποιούνται εμφωλευμένα ερωτήματα πολλαπλών επιπέδων. Το ερώτημα του πρώτου επιπέδου επιστρέφει λίστα με όλους τους (μη μηδενικούς) λόγους αξία/έκταση, ένα για κάθε αγρόκτημα. Το ερώτημα του δεύτερου επιπέδου επιλέγει την μικρότερη τιμή της στήλης dols_per_acre. Τέλος το εξωτερικό ερώτημα επιστρέφει τον κωδικό του αγροκτήματος και την αξία του ανά στρέμμα:

Data Output		Explain	Messages	Notifications
	parcel_id integer	dols_per_acre double precision		
1	5567	281.145742773064		

Έξοδος explain:



MX4. Χρήση των τελεστών συνάθροισης AVG() και COUNT(). Χρήση εμφωλευμένου ερωτήματος.

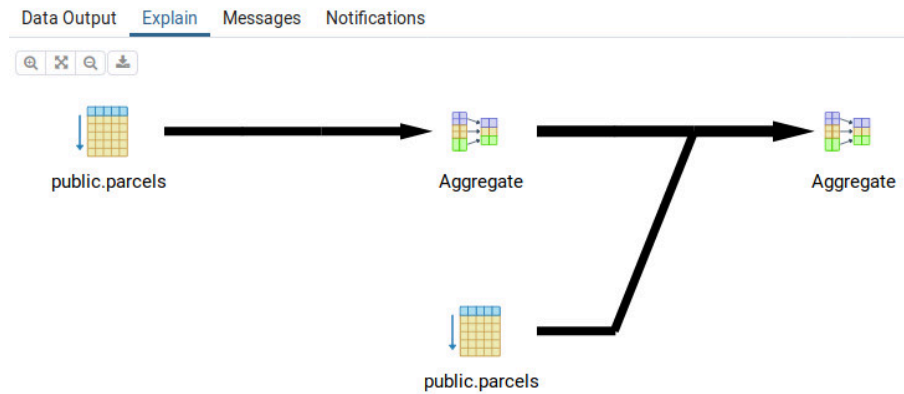
Ερώτημα: Πόσα είναι τα αγροκτήματα των οποίων η αξία ανά στρέμμα (μη μηδενική) είναι μεγαλύτερη από το μέσο όρο.

```
SELECT COUNT(*)
FROM parcels
WHERE asmt/acres > (SELECT AVG(T.dols_per_acre)
                    FROM(SELECT asmt/acres AS dols_per_acre
                        FROM parcels
                        WHERE asmt/acres <> 0) AS T)
```

Χρησιμοποιούνται εμφωλευμένα ερωτήματα πολλαπλών επιπέδων. Το ερώτημα του πρώτου επιπέδου επιστρέφει λίστα με όλους τους (μη μηδενικούς) λόγους αξία/έκταση, ένα για κάθε αγρόκτημα. Το ερώτημα του δεύτερου επιπέδου υπολογίζει το μέσο όρο της στήλης dols_per_acre. Τέλος το εξωτερικό ερώτημα καταμετρά και επιστρέφει το πλήθος των αγροκτημάτων των οποίων η αξία είναι μεγαλύτερη από αυτόν τον μέσο όρο:

Data Output		Explain	Messages	Notifications
	count bigint			
1	154			

Έξοδος explain:



MX5. Χρήση ερωτήματος σύνδεσης JOIN. Χρήση του τελεστή συνάθροισης COUNT() σε στοιχεία που έχουν ομαδοποιηθεί με τον τελεστή ομαδοποίησης GROUP_BY(). Χρήση εμφωλευμένου ερωτήματος.

Ερώτημα: Πόσα είναι τα αγροκτήματα που ασχολούνται με την καλλιέργεια του κάθε φυτού κάθε εποχή του χρόνου.

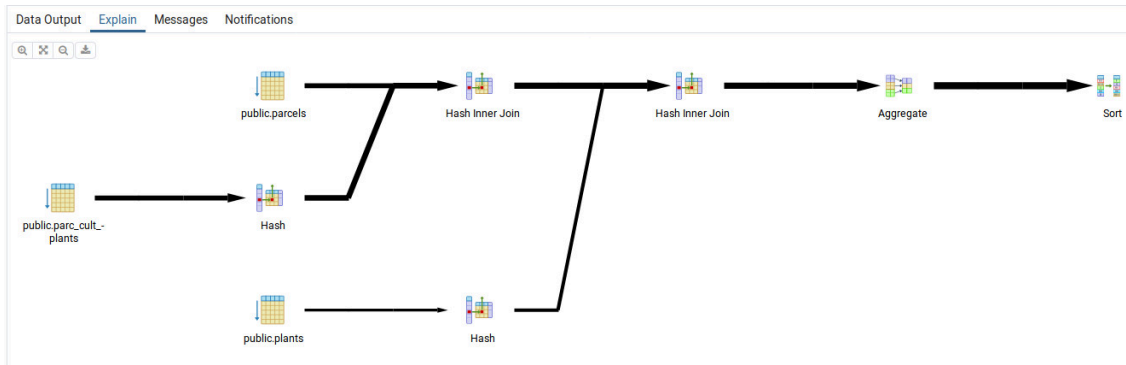
```

SELECT p.season, pcp.plant_id, COUNT(*) AS num_of_parcels
FROM parcels pc JOIN parc_cult_plants pcp ON pc.parcel_id =
      pcp.parcel_id JOIN plants p ON pcp.plant_id = p.name
GROUP BY pcp.plant_id, p.season
ORDER BY p.season, pcp.plant_id ASC
  
```

Χρησιμοποιώ την συνένωση (JOIN) 3 συνολικά πινάκων που διαθέτουν τα κατάλληλα στοιχεία. Το ερώτημα τελικά επιστρέφει:

	season character varying (10)	plant_id character varying (20)	num_of_parcels bigint
1	Autumn	plant_14	1165
2	Spring	plant_10	1159
3	Spring	plant_11	1116
4	Spring	plant_13	1110
5	Spring	plant_4	1109
6	Spring	plant_7	1156
7	Spring	plant_8	1165
8	Summer	plant_1	1128
9	Summer	plant_12	1156
10	Summer	plant_3	1096
11	Summer	plant_9	1188
12	Winter	plant_15	1142
13	Winter	plant_2	1117
14	Winter	plant_5	1089
15	Winter	plant_6	1168

Έξοδος explain:



MX6. Χρήση ερωτήματος σύνδεσης JOIN. Χρήση του τελεστή συνάθροισης AVG() σε στοιχεία που έχουν ομαδοποιηθεί με τον τελεστή ομαδοποίησης GROUP_BY(). Χρήση εμφωλευμένου ερωτήματος.

Ερώτημα: Ποια εποχή ευδοκίμει το φυτό με την καλύτερη απόδοση καλλιέργειας.

```
SELECT season
FROM (
    SELECT ap.plant_id, p.season, AVG(performance) AS avg_perf
    FROM plants p JOIN annual_performances ap ON p.name =
    ap.plant_id
    GROUP BY ap.plant_id, p.season
    ORDER BY AVG(performance) DESC
) AS T
LIMIT 1
```

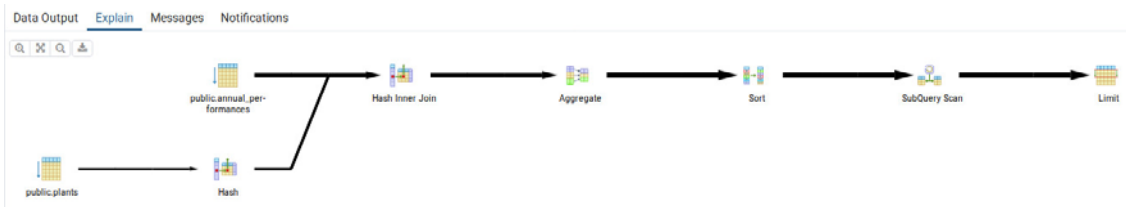
Το εμφωλευμένο ερώτημα εκτελεί την συνένωση (JOIN) 2 συνολικά πινάκων που διαθέτουν τα κατάλληλα στοιχεία και επιστρέφει ταξινομημένη λίστα με τον κωδικό του κάθε φυτού, την εποχή ευδοκίμησης και την αντίστοιχη απόδοση.

	plant_id character varying (20)	season character varying (10)	avg_perf numeric
1	plant_13	Spring	559.9877942458587620
2	plant_4	Spring	558.6444249341527656
3	plant_15	Winter	556.8500443655723159
4	plant_12	Summer	556.0407986111111111
5	plant_10	Spring	553.9589652096342551
6	plant_6	Winter	552.1053097345132743
7	plant_9	Summer	551.760661447345518

Το εξωτερικό ερώτημα επιλέγει το πρώτο στοιχείο της λίστας και παρουσιάζει τα στοιχεία της αντίστοιχης εποχής (season):

	season character varying (10)
1	Spring

Έξοδος explain:



MX7. Χρήση εμφωλευμένου ερωτήματος πολλαπλών επιπέδων. Χρήση του τελεστή συνάθροισης AVG() και του τελεστή συνάθροισης SUM() σε στοιχεία που έχουν ομαδοποιηθεί με τον τελεστή ομαδοποίησης GROUP_BY(). Χρήση του τελεστή IN() για να αποδοθούν πολλαπλές συγκεκριμένες τιμές στο πεδίο WHERE. Χρήση του τελεστή :: για αλλαγή του τύπου δεδομένων. Χρήση του τελεστή χαρακτήρα Left() για να απομονωθεί ένα συγκεκριμένο αριστερό τμήμα ενός string.

Ερώτημα: Να υπολογιστεί η μέση απόδοση καλλιέργειας του φυτού plant_15 στα αγροκτήματα που το id τους ξεκινά με '100'.

```

SELECT AVG(sum_perf) AS average_performance
FROM (SELECT parcel_id, SUM(performance) AS sum_perf
      FROM annual_performances
      WHERE parcel_id IN (SELECT parcel_id
                        FROM parcels
                        WHERE Left(parcel_id::text,3) = '100'
                        )
      AND plant_id = 'plant_15'
      GROUP BY parcel_id) AS T
  
```

Χρησιμοποιούνται εμφωλευμένα ερωτήματα πολλαπλών επιπέδων. Το ερώτημα του πρώτου επιπέδου επιστρέφει λίστα με όλους τους κωδικούς αγροκτημάτων που ξεκινάνε με '100'. Το ερώτημα του δεύτερου επιπέδου υπολογίζει την απόδοση για κάθε ένα από τα παραπάνω αγροκτήματα, αναφορικά με το ζητούμενο φυτό. Τέλος το εξωτερικό ερώτημα υπολογίζει και επιστρέφει το μέσο όρο απόδοσης όλων των αγροκτημάτων στο ζητούμενο φυτό:

Data Output	Explain	Messages	Notifications
	average_performance numeric		
1	863.0000000000000000		

Έξοδος explain:



MX8. Δημιουργία και χρήση View για την προσωρινή αποθήκευση χρήσιμων στοιχείων. Χρήση του τελεστή συνάθροισης COUNT() σε στοιχεία που έχουν ομαδοποιηθεί με τον τελεστή ομαδοποίησης GROUP_BY(). Χρήση του τελεστή IN() για να αποδοθούν πολλαπλές συγκεκριμένες τιμές στο πεδίο WHERE. Χρήση εμφωλευμένου ερωτήματος.

Ερώτημα: Να υπολογιστεί το πλήθος των ατόμων που δουλεύουν σε κάθε ένα από τα 5 αγροκτήματα, που παρουσιάζουν τις μεγαλύτερες ετήσιες αποδόσεις.

MX8.1 Δημιουργία όψης όπου αποθηκεύονται τα στοιχεία (id και απόδοση) των 5 πιο αποδοτικών αγροκτημάτων

```
-- DROP VIEW max_perf CASCADE;
CREATE VIEW max_perf AS
  SELECT parcel_id, SUM(performance) AS total_performance
  FROM annual_performances
  GROUP BY parcel_id
  ORDER BY SUM(performance) DESC
  LIMIT 5;
-- SELECT * FROM max_perf
```

MX8.2 Ανάκτηση των τελικών στοιχείων του ερωτήματος με χρήση των στοιχείων που είναι αποθηκευμένα στην παραπάνω όψη.

```
SELECT parcel_id, COUNT(*)
FROM people
WHERE parcel_id IN (SELECT parcel_id
                    FROM max_perf)
GROUP BY parcel_id
```

Data Output	Explain	Messages	Notifications
parcel_id integer	count bigint		
1	151	2	
2	1404	3	
3	1748	1	
4	3614	1	
5	5524	2	

Έξοδος explain:



6. ΧΩΡΙΚΑ ΕΡΩΤΗΜΑΤΑ

Θα εκτελέσω σειρά από χωρικά ερωτήματα με χρήση τοπολογικών τελεστών, όπως αυτοί που ζητούνται στην εκφώνηση. Οι γεωμετρίες των επιστρεφόμενων από τα ερωτήματα γεωμετριών εμφανίζονται στο χάρτη με ΓΑΛΑΖΙΟ ή ΚΟΚΚΙΝΟ κατά περίπτωση χρώμα. Στα περισσότερα των ερωτημάτων τα αποτελέσματα οδηγούνται σε έναν καινούργιο πίνακα, τον `query_layer`, ο οποίος χρησιμεύει για την οπτικοποίηση των αποτελεσμάτων στο qGIS.

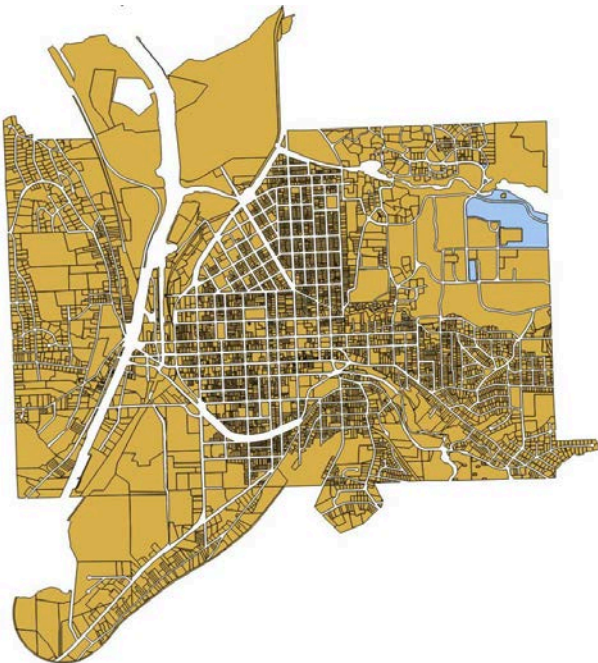
X1. Υπολογισμός εμβαδού, με τη χρήση του τοπολογικού τελεστή `ST_Area()`. Χρήση του τελεστή `ST_Union()` για την ενοποίηση γεωμετριών.

Ερώτημα: Ζητείται ο υπολογισμός του συνολικού εμβαδού των 5 πιο ακριβών σε αξία κτημάτων (`parcels`) του χάρτη.

Η συνολική αξία ενός κτήματος (αξία γης και αξία κτισμάτων) περιέχεται στην στήλη `asmt` (`assessment`) του πίνακα `parcels`.

X1.1 Αρχικά εκτελώ βοηθητικό ερώτημα ώστε να εξάγω τις γεωμετρίες των ζητούμενων 5 πιο ακριβών κτημάτων, για να μπορέσω να τις εμφανίσω στο χάρτη:

```
DROP TABLE query_layer;  
SELECT parcels.geometry  
INTO query_layer  
FROM parcels  
ORDER BY parcels.asmt DESC  
LIMIT 5
```



X1.2 Στη συνέχεια εκτελώ βοηθητικό ερώτημα που επιστρέφει τα επιμέρους εμβαδά των 5 ζητούμενων κτημάτων, ώστε να μπορέσω να ελέγξω το τελικό αποτέλεσμα:

```
SELECT ST_Area(parcels.geometry)  
FROM parcels  
ORDER BY parcels.asmt DESC  
LIMIT 5
```

Data Output	Explain	Messages	Notifications	Geometry Viewer
st_area double precision				
1	90623.4880956653			
2	274179.086266543			
3	1386626.70290601			
4	3875.18675778671			
5	9726.07678012114			

X1.3 Εκτέλεση του κυρίως ερωτήματος. Θα μπορούσα απλά να αθροίσω τα 5 επιμέρους εμβαδά, που επιστρέφει το προηγούμενο ερώτημα. Αντίθετα, θα εκμεταλλευτώ τον τελεστή ST_Union(), ώστε να ενοποιήσω τις επιμέρους γεωμετρίες σε μία ενιαία, της οποίας στο τέλος υπολογίζω το συνολικό εμβαδόν. Για την εκτέλεση του ST_Union() πρέπει οι επιμέρους γεωμετρίες να εμφανιστούν με την μορφή λίστας (array):

```
SELECT ST_Area(
    (SELECT ST_UNION(
        ARRAY(
            SELECT parcels.geometry
            FROM parcels
            ORDER BY parcels.asmt DESC
            LIMIT 5
        )
    )
)
```

Data Output	Explain	Messages	Notifications	Geometry Viewer
st_area double precision				
1	1765030.54080612			

Η επιστρεφόμενη τιμή ισούται με το άθροισμα των 5 επιμέρους εμβαδών.

Έξοδος explain:

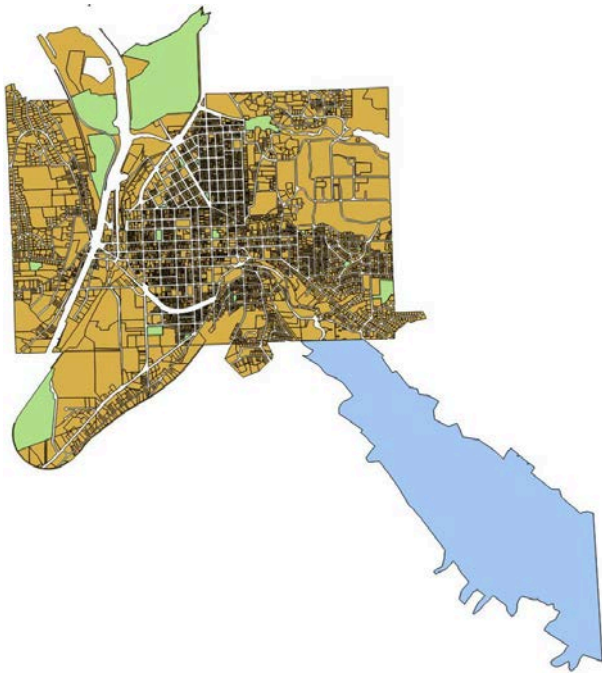


X2. Υπολογισμός περιμέτρου, με τη χρήση του τοπολογικού τελεστή ST_Perimeter(). Χρήση εμφωλευμένου ερωτήματος.

Ερώτημα: Ζητείται ο υπολογισμός της περιμέτρου του πιο μεγάλου πάρκου του χάρτη.

X2.1 Αρχικά εκτελώ βοηθητικό ερώτημα, ώστε να εξάγω το όνομα και τη γεωμετρία του ζητούμενου πάρκου, για να μπορέσω να το εμφανίσω στο χάρτη:

```
DROP TABLE query_layer;
SELECT name, parks.geometry
INTO query_layer
FROM parks
ORDER BY parks.size DESC
LIMIT 1
```

X2.2 Εκτέλεση του κυρίως ερωτήματος. Υπολογίζω την περίμετρο της επιστρεφόμενης γεωμετρίας:

```
SELECT ST_Perimeter(T.geometry)
FROM(
    SELECT parks.geometry
    FROM parks
    ORDER BY parks.size DESC
    LIMIT 1) AS T
```

Data Output	Explain	Messages	Notifications	Geometry Viewer
st_perimeter				
double precision				
1	48197.2941853321			

Έξοδος explain:



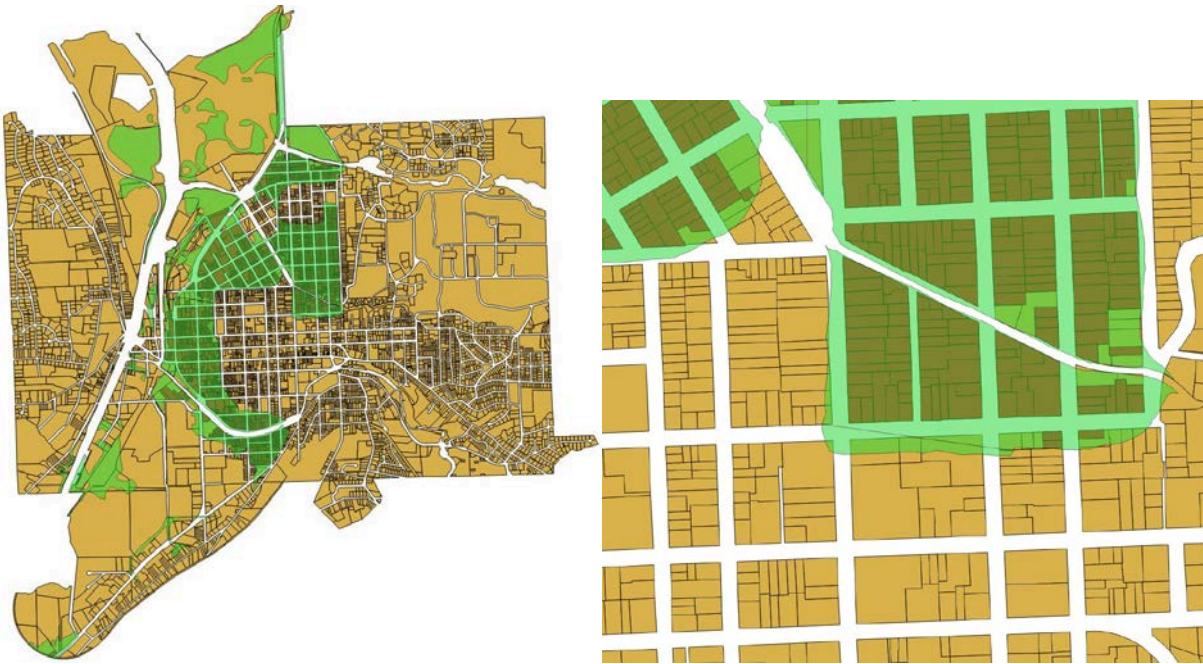
Στα επόμενα 3 ερωτήματα θα προσπαθήσω να αναδείξω τις χαρακτηριστικές διαφορές των αποτελεσμάτων των τοπολογικών τελεστών ST_Contains(), ST_Intersects() και ST_Overlaps().

X3. Υπολογισμός γεωμετριών που εμπεριέχονται πλήρως μέσα σε μία άλλη γεωμετρία, με τη χρήση του τοπολογικού τελεστή ST_Contains().

Ερώτημα: Ζητείται ο εντοπισμός των κτημάτων που καλύπτονται πλήρως από τη ζώνη X500 του firm.

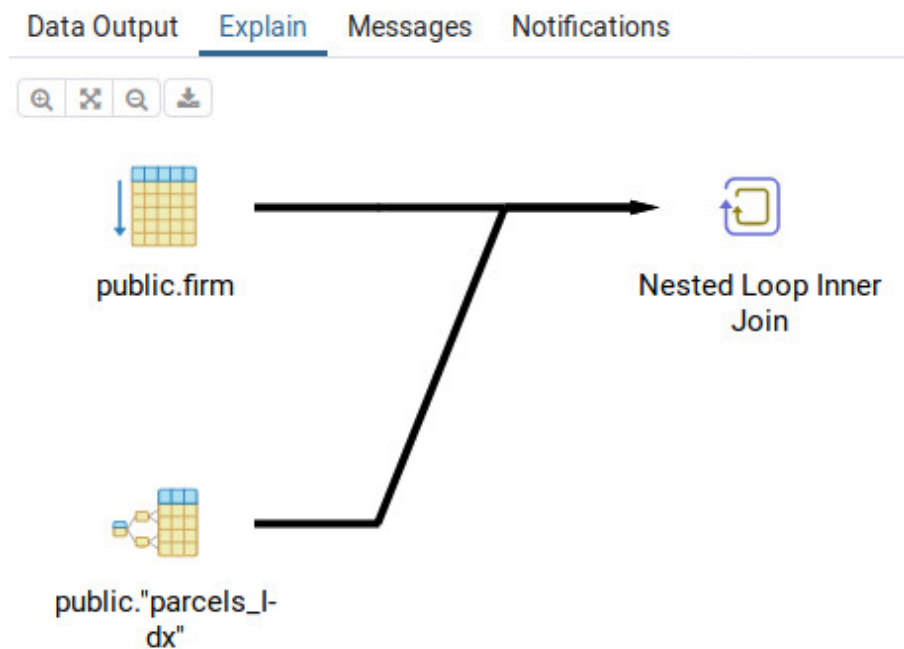
```
DROP TABLE query_layer;
SELECT parcels.*
INTO query_layer
FROM parcels, firm
WHERE ST_Contains(firm.geometry, parcels.geometry)
```

AND firm.zone = 'X500'



Όπως φαίνεται, τόσο στο συνολικό χάρτη αριστερά όσο και στο μεγεθυμένο τμήμα δεξιά, οι επιστρεφόμενες από το ερώτημα γεωμετρίες καλύπτονται πλήρως από την εξεταζόμενη ζώνη (ανοιχτό πράσινο χρώμα).

Έξοδος explain:

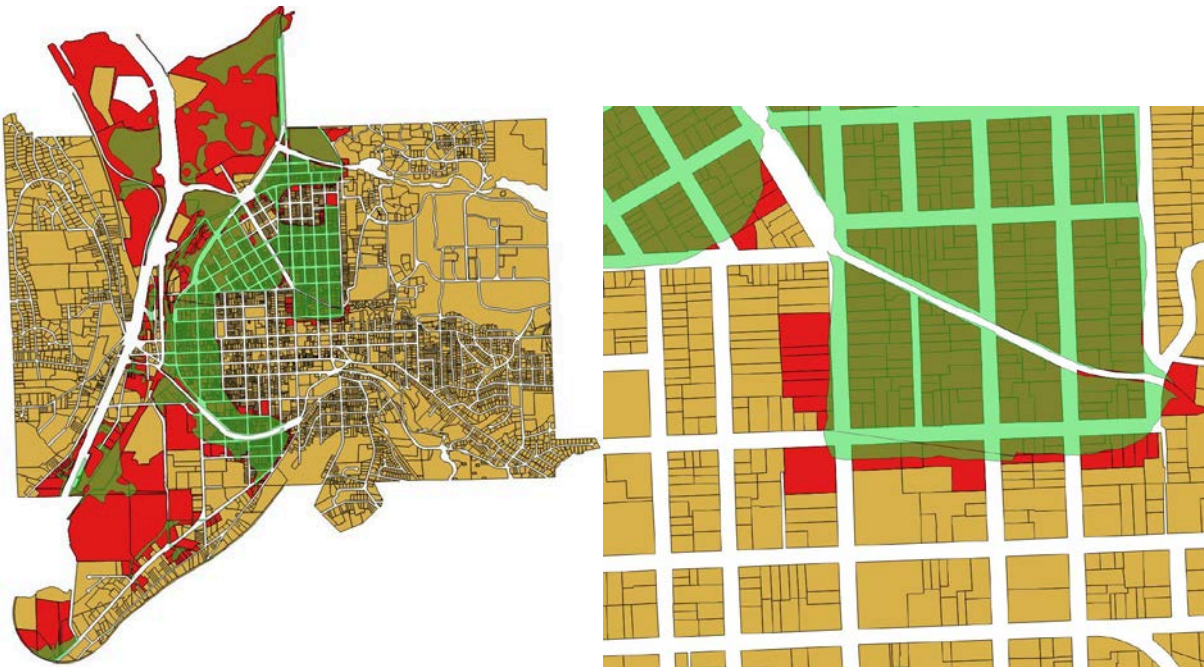


X4. Υπολογισμός γεωμετριών που τέμνονται (δεν είναι αναγκαίο να καλύπτονται πλήρως) από μία άλλη γεωμετρία, με τη χρήση του τοπολογικού τελεστή ST_Intersects().

Ερώτημα: Ζητείται ο εντοπισμός των κτημάτων που τέμνονται από τη ζώνη X500 του firm.

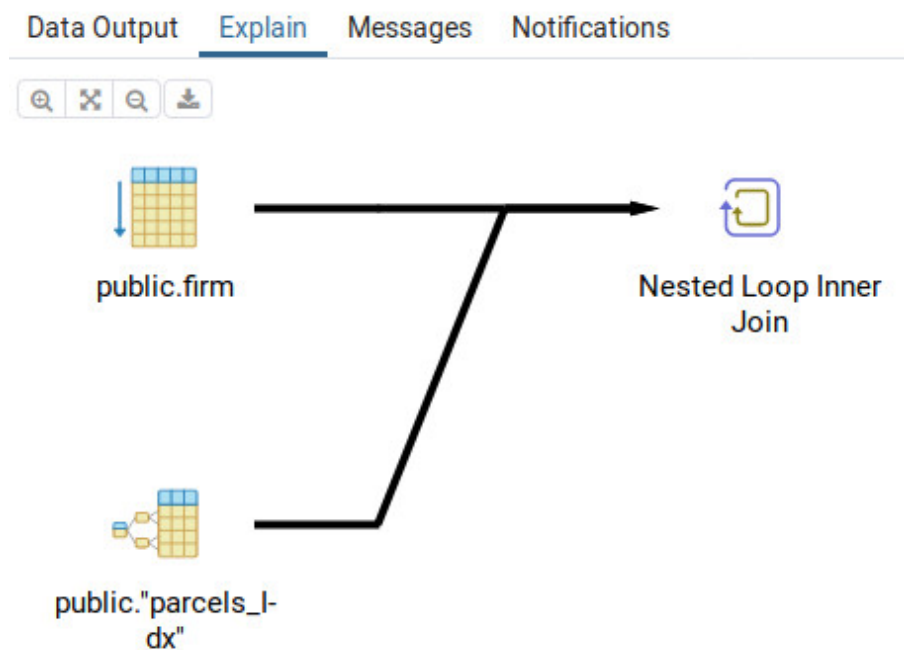
```
DROP TABLE query_layer;
```

```
SELECT parcels.*  
INTO query_layer  
FROM parcels, firm  
WHERE ST_Intersects(firm.geometry, parcels.geometry)  
      AND firm.zone = 'X500'
```



Σε σχέση με το προηγούμενο, στο παρόν ερώτημα επιστρέφονται επιπρόσθετα και οι γεωμετρίες που καλύπτονται μερικώς από την εξεταζόμενη ζώνη.

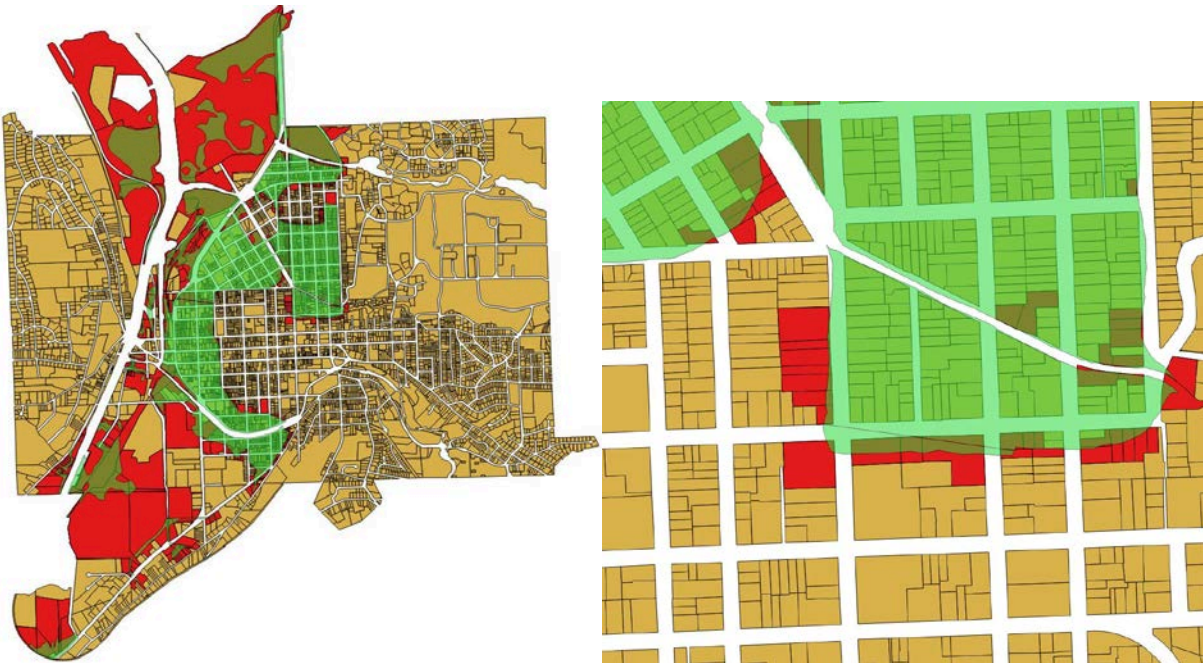
Έξοδος explain:



X5. Υπολογισμός γεωμετριών που επικαλύπτονται (απλά ένα μέρος τους, όχι πλήρως) από μία άλλη γεωμετρία, με τη χρήση του τοπολογικού τελεστή `ST_Overlaps()`.

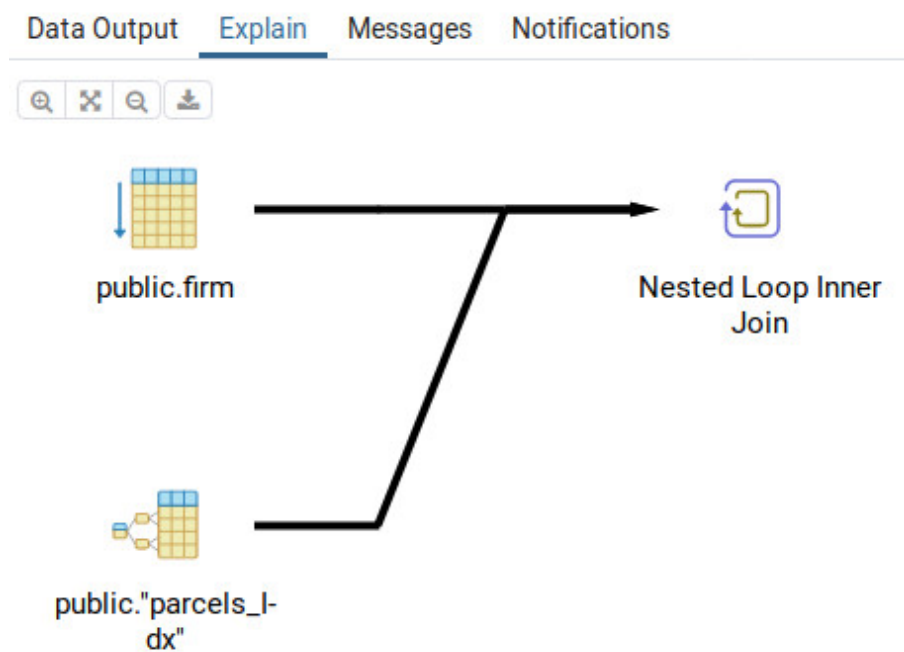
Ερώτημα: Ζητείται ο εντοπισμός των κτημάτων που επικαλύπτονται από τη ζώνη X500 του firm.

```
DROP TABLE query_layer;  
SELECT parcels.*  
INTO query_layer  
FROM parcels, firm  
WHERE ST_Overlaps(firm.geometry, parcels.geometry)  
      AND firm.zone = 'X500'
```



Επιστρέφονται οι γεωμετρίες που καλύπτονται μόνο μερικώς από την εξεταζόμενη ζώνη

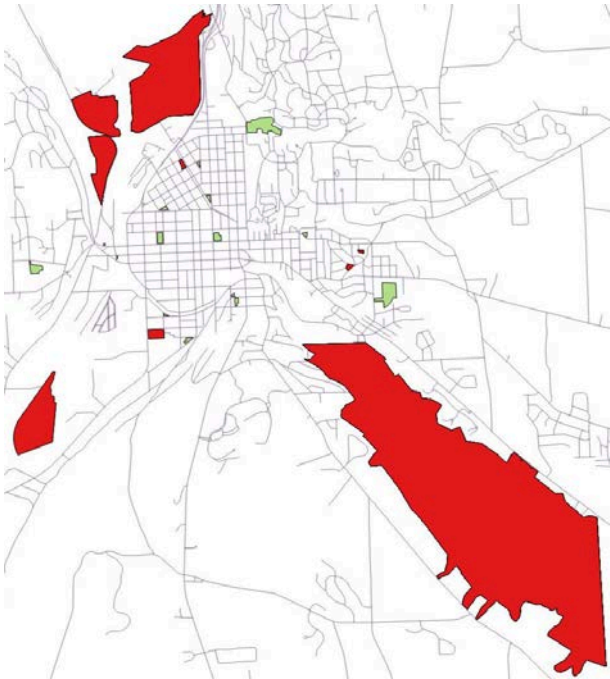
Έξοδος explain:



X6. Υπολογισμός γεωμετριών που διαπερνούνται (έχουν κάποια από τα εξωτερικά τους σημεία κοινά, αλλά όχι όλα) από μία άλλη γεωμετρία, με τη χρήση του τοπολογικού τελεστή ST_Crosses().

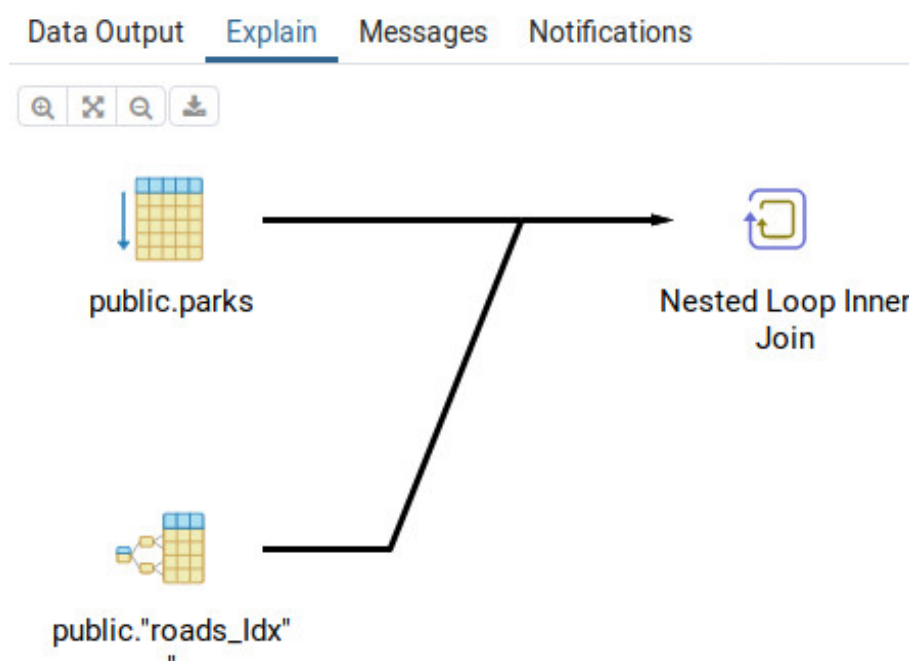
Ερώτημα: Ζητείται ο εντοπισμός των πάρκων από τα οποία διέρχεται κάποιος δρόμος.

```
DROP TABLE query_layer;  
SELECT parks.geometry  
INTO query_layer  
FROM parks, roads  
WHERE ST_Crosses(parks.geometry,roads.geometry)
```



Τα ζητούμενα πάρκα τα οποία διατρέχονται από κάποιο δρόμο εμφανίζονται με κόκκινο χρώμα.

Έξοδος explain:



X7. Υπολογισμός απόστασης ανάμεσα σε γεωμετρίες, με τη χρήση του τοπολογικού τελεστή ST_Distance(). Στη συγκεκριμένη περίπτωση θα γίνει χρήση ενός βοηθητικού εμφωλευμένου ερωτήματος, που ζητείται από την εκφώνηση της εργασίας.

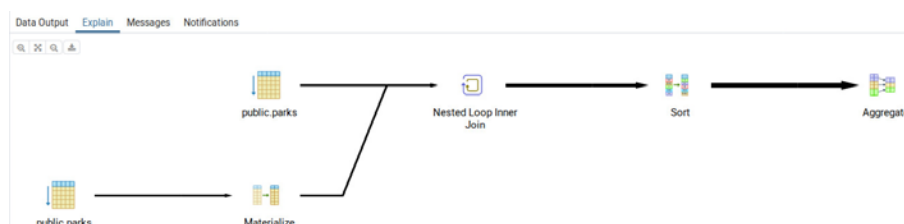
Ερώτημα: Ζητείται ο υπολογισμός της μέγιστης απόστασης, που μπορεί να εμφανίζει το κάθε πάρκο από κάποιο άλλο πάρκο.

```
SELECT park1_name, MAX(parks_dist) AS max_parks_dist
FROM (
    SELECT p1.name AS park1_name, p2.name AS park2_name,
           ST_Distance(p1.geometry, p2.geometry) AS parks_dist
    FROM parks AS p1, parks AS p2
    WHERE p1.name <> p2.name
    ORDER BY p1.name, parks_dist ASC
) AS PD
GROUP BY park1_name
```

Το ερώτημα επιστρέφει τον παρακάτω πίνακα όπου εμφανίζεται τα ονόματα των πάρκων καθώς και η μέγιστη απόσταση του καθενός από αυτά από κάποιο άλλο πάρκο.

	park1_name character varying (30)	max_parks_dist double precision
1	Auburn Park	9129.87045401456
2	Baker Park	7728.7527761361
3	Brindley Park	9999.31214819128
4	Bryant Park	11277.7782503071
5	Cass Park	10522.0821707127
6	Columbia Street Park	6990.02677413086
7	Conley Park	8709.19609988305
8	Conway Park	8178.06551701259
9	Dewitt Park	7506.0737607735
10	Dryden Rd Park	10329.4047715694
11	Hillview Park	7067.11801197276
12	Ithaca Falls Natural Area	11184.235051962
13	Maplewood Park	11883.181717328
14	McDaniels Park	12128.1379915094
15	Negundo Woods	11926.2382518551
16	Six Mile Creek Natural Ar...	9854.12995036707
17	Stewart Park	9293.20184484476
18	Strawberry Fields	12128.1379915094
19	Thompson Park	8365.77998095679
20	Titus Triangle	7501.47553947362
21	Van Horn Park	9474.2767851747
22	Washington Park	7963.02811687143
23	Wood Street Park	7787.04021295716

Έξοδος explain:



X8. Υπολογισμός απόστασης ανάμεσα σε γεωμετρίες, με τη χρήση του τοπολογικού τελεστή ST_Distance(). Στη συγκεκριμένη περίπτωση θα συντελεστεί δημιουργία και χρήση όψης, που ζητείται από την εκφώνηση της εργασίας. Χρήση του τελεστή συνάθροισης MIN().

Ερώτημα: Ζητείται ο υπολογισμός της ελάχιστης απόστασης, που εμφανίζει το κάθε πάρκο από κάποιο άλλο πάρκο.

X8.1 Δημιουργία View που περιέχει τις αποστάσεις του κάθε πάρκου από κάθε άλλο πάρκο.

```
-- DROP VIEW Parks_Distances CASCADE;
CREATE VIEW Parks_Distances AS
  SELECT  p1.name AS park1_name,  p2.name AS park2_name,
          ST_Distance(p1.geometry, p2.geometry) AS parks_dist
  FROM parks AS p1, parks AS p2
  WHERE p1.name <> p2.name
  ORDER BY p1.name, parks_dist ASC;
-- SELECT * FROM Parks_Distances
```

X8.2 Δημιουργία View που περιέχει το όνομα και την ελάχιστη απόσταση του κάθε πάρκου από κάποιο άλλο πάρκο. Αξιοποιώ τα αποτελέσματα του παραπάνω View.

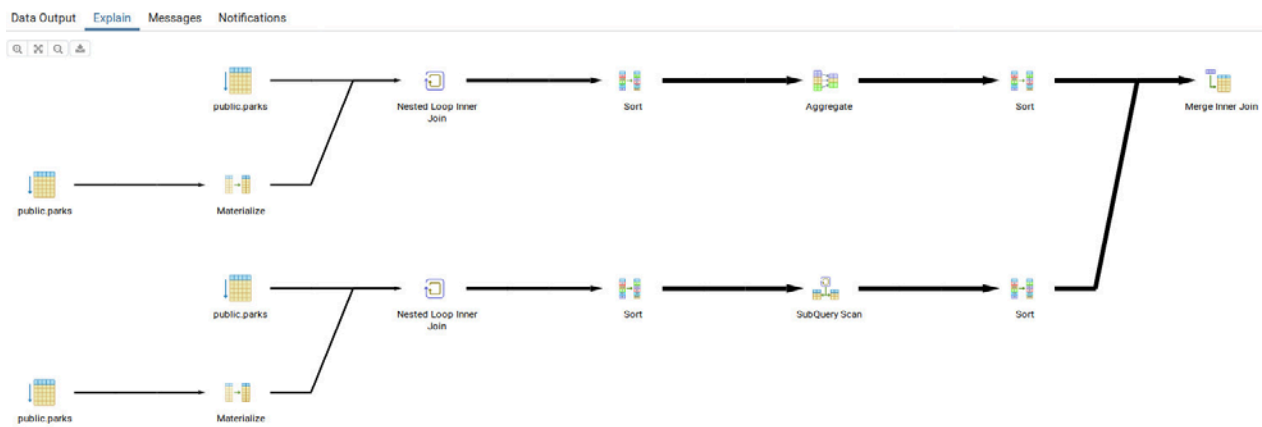
```
-- DROP VIEW Min_Parks_Distance CASCADE;
CREATE VIEW Min_Parks_Distance AS
  SELECT park1_name, MIN(parks_dist) AS min_parks_dist
  FROM Parks_Distances
  GROUP BY park1_name;
-- SELECT * FROM Min_Parks_Distance
```

X8.3 Αν και το τελευταίο View επιστρέφει τα στοιχεία που ζητούνται από το ερώτημα, εκτελώ ακόμα ένα ερώτημα, ώστε να εμφανίζεται στη δεύτερη στήλη και το όνομα του πάρκου που απέχει την μικρότερη απόσταση από το πάρκο της πρώτης στήλης. Γίνεται συνδυασμένη χρήση των αποτελεσμάτων των δύο παραπάνω Views.

```
SELECT minpd.park1_name, pd.park2_name, minpd.min_parks_dist
FROM Min_Parks_Distance AS minpd, Parks_Distances AS pd
WHERE minpd.min_parks_dist = pd.parks_dist
  AND minpd.park1_name <> pd.park2_name
ORDER BY minpd.min_parks_dist
```

	park1_name character varying (30)	park2_name character varying (30)	min_parks_dist double precision
1	Hillview Park	Columbia Street Park	50.3136510213305
2	Columbia Street Park	Hillview Park	50.3136510213305
3	Stewart Park	Cass Park	404.135660172029
4	Cass Park	Stewart Park	404.135660172029
5	Conley Park	Auburn Park	416.758682732022
6	Auburn Park	Conley Park	416.758682732022
7	Wood Street Park	Titus Triangle	475.005929432781
8	Titus Triangle	Wood Street Park	475.005929432781
9	Bryant Park	Maplewood Park	509.601452290296
10	Maplewood Park	Bryant Park	509.601452290296
11	Brindley Park	Van Horn Park	572.521132624569
12	Van Horn Park	Brindley Park	572.521132624569
13	Baker Park	Wood Street Park	737.700824016001
14	Conway Park	Washington Park	803.976406964915
15	Washington Park	Conway Park	803.976406964915
16	Thompson Park	Auburn Park	1028.59647424951
17	Dewitt Park	Thompson Park	1078.85002902921
18	Strawberry Fields	Bryant Park	1207.8666757962
19	Six Mile Creek Natural Ar...	Strawberry Fields	1432.64186531531
20	Dryden Rd Park	Bryant Park	1779.97610933231
21	Ithaca Falls Natural Area	Stewart Park	1856.97848007439
22	McDaniels Park	Brindley Park	2350.64891411271
23	Negundo Woods	Wood Street Park	3614.46986701281

Έξοδος explain:



7. ΠΑΡΑΡΤΗΜΑ

Στην παρούσα ενότητα θα καταγραφεί όλος ο κώδικας που χρησιμοποιήθηκε σε διάφορα στάδια της εργασίας. Ο κώδικας επίσης περιέχεται και στα αντίστοιχα αρχεία του φακέλου 'Συνοδευτικά αρχεία'.

1. Κώδικας SQL για την αφαίρεση από την αρχική δομή της ΒΔ των πινάκων που δεν θα χρησιμοποιηθούν (αρχείο *'Drop unnecessary tables.sql'*).

```
DROP TABLE ctab, dem, elevpts, leftsquare, middle, parkrast,
parks_utm, parvalues, propclas, qlayer, rasslope, rightsquare,
treetable, upstate;
```

2. Κώδικας SQL για τη δημιουργία όλων των απαραίτητων επιπλέον πινάκων ΒΔ (αρχείο *'Creation of necessary tables.sql'*).

```
-- Table: public.people

-- DROP TABLE public.people;

CREATE TABLE public.people
(
    name character varying(20) COLLATE pg_catalog."default" NOT NULL,
    parcel_id integer,
    CONSTRAINT people_pkey PRIMARY KEY (name),
    CONSTRAINT parcelid FOREIGN KEY (parcel_id)
        REFERENCES public.parcels ("OID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

-- Table: public.plants

-- DROP TABLE public.plants;

CREATE TABLE public.plants
(
    name character varying(20) COLLATE pg_catalog."default" NOT NULL,
    season character varying(10) COLLATE pg_catalog."default",
    CONSTRAINT plants_pkey PRIMARY KEY (name)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

-- Table: public.annual_performances

-- DROP TABLE public.annual_performances;
```

```
CREATE TABLE public.annual_performances
(
    ap_id bigint NOT NULL,
    parcel_id integer,
    plant_id character varying(20) COLLATE pg_catalog."default",
    performance integer,
    CONSTRAINT annual_performances_pkey PRIMARY KEY (ap_id),
    CONSTRAINT parcelid FOREIGN KEY (parcel_id)
        REFERENCES public.parcels ("OID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT plantid FOREIGN KEY (plant_id)
        REFERENCES public.plants (name) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

-- Table: public.parc_cult_plants

-- DROP TABLE public.parc_cult_plants;

CREATE TABLE public.parc_cult_plants
(
    parcel_id integer NOT NULL,
    plant_id character varying(20) COLLATE pg_catalog."default" NOT
NULL,
    CONSTRAINT parc_cult_plants_pkey PRIMARY KEY (parcel_id,
plant_id),
    CONSTRAINT parcelid FOREIGN KEY (parcel_id)
        REFERENCES public.parcels ("OID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT plantid FOREIGN KEY (plant_id)
        REFERENCES public.plants (name) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

-- Firm tables for various zone values
SELECT geometry
INTO firm_A
FROM firm
WHERE firm.zone = 'A'

SELECT geometry
INTO firm_AE
FROM firm
```



```
WHERE firm.zone = 'AE'

SELECT geometry
INTO firm_X
FROM firm
WHERE firm.zone = 'X'

SELECT geometry
INTO firm_X500
FROM firm
WHERE firm.zone = 'X500'
```

3. Κώδικας Python για τη δημιουργία και αποθήκευση σε κατάλληλα αρχεία .csv συνθετικών μη χωρικών δεδομένων για να συμπληρώσουν με τους μη χωρικούς πίνακες (αρχείο *'Populate tables.py'*).

```
import pandas as pd
import numpy as np
import csv

# Primary keys from parcel's table
parsel_id = pd.read_csv('parcels_id.csv', header=None).iloc[:,0]

## Create data for people table
# I will create 10.000 people and assign them to random parsel_id
# data must be at the form (people_name, parsel_id)
id_choices = np.random.choice(parsel_id, size=10000, replace=True)
peoples = []
for i in range(10001):
    if i == 0:
        continue
    name = 'people_'+str(i)
    peoples.append([name,id_choices[i-1]])

# Save data as .csv file
with open("peoples.csv", "w") as f:
    writer = csv.writer(f)
    writer.writerows(peoples)

## Create data for plants table
# I will create 15 plants. Data must be at the form (plant_name,
season)
seasons = ['Winter', 'Spring', 'Summer', 'Autumn']
season_choices = list(np.random.choice(seasons, size=15,
replace=True))
plants_names = []
plants = []
for i in range(16):
    if i == 0:
        continue
    plant = 'plant_'+str(i)
    plants_names.append(plant)
    plants.append([plant,str(season_choices[i-1])])
```

```
# Save data as .csv file
with open("plants.csv", "w") as f:
    writer = csv.writer(f)
    writer.writerows(plants)

## Create data for annual_performances table
# For each parcel I will choose 3 random plants and a random
performance for
# each plant. Data must be at the form (id, parcel_id, plant_id,
performance)
an_perf = []
idx = 1
for parcel in parcel_id:
    plants = list(np.random.choice(plants_names, size=3,
replace=False))
    for plant in plants:
        perf = np.random.choice(range(100,1000), size=1)
        an_perf.append([idx,parcel,str(plant),perf[0]])
        idx+=1

# Save data as .csv file
with open("annual_performance.csv", "w") as f:
    writer = csv.writer(f)
    writer.writerows(an_perf)

## Create data for parc_cult_plants table
# For each parcel I will choose 3 random plants. Data must be at the
form
# (parcel_id, plant_id)
p_c_p = []
for parcel in parcel_id:
    plants = list(np.random.choice(plants_names, size=3,
replace=False))
    for plant in plants:
        p_c_p.append([parcel,str(plant)])

# Save data as .csv file
with open("parc_cult_plants.csv", "w") as f:
    writer = csv.writer(f)
    writer.writerows(p_c_p)
```

4. Ο κώδικας SQL των μη χωρικών ερωτημάτων (αρχείο 'Non Spatial Queries.sql').

```
-- NON SPATIAL QUERY MX1
SELECT parcel_id, SUM(performance) AS total_performance
FROM annual_performances
GROUP BY parcel_id
ORDER BY SUM(performance) DESC

-- NON SPATIAL QUERY MX2
SELECT MAX(employees)
FROM(
    SELECT parcel_id, COUNT(*) AS employees
    FROM people
```

```
GROUP BY parcel_id
ORDER BY COUNT(*) DESC
) AS T

-- NON SPATIAL QUERY MX3
SELECT parcel_id, asmt/acres AS dols_per_acre
FROM parcels
WHERE asmt/acres = (SELECT MIN(T.dols_per_acre)
                    FROM (SELECT asmt/acres AS dols_per_acre
                          FROM parcels
                          WHERE asmt/acres <> 0) AS T)

-- NON SPATIAL QUERY MX4
SELECT COUNT(*)
FROM parcels
WHERE asmt/acres > (SELECT AVG(T.dols_per_acre)
                    FROM (SELECT asmt/acres AS dols_per_acre
                          FROM parcels
                          WHERE asmt/acres <> 0) AS T)

-- NON SPATIAL QUERY MX5
SELECT p.season, pcp.plant_id, COUNT(*) AS num_of_parcel
FROM parcels pc JOIN parc_cult_plants pcp ON pc.parcel_id =
pcp.parcel_id JOIN plants p ON pcp.plant_id = p.name
GROUP BY pcp.plant_id, p.season
ORDER BY p.season, pcp.plant_id ASC

-- NON SPATIAL QUERY MX6
SELECT season
FROM (
  SELECT ap.plant_id, p.season, AVG(performance) AS avg_perf
  FROM plants p JOIN annual_performances ap ON p.name = ap.plant_id
  GROUP BY ap.plant_id, p.season
  ORDER BY AVG(performance) DESC
) AS T
LIMIT 1

-- NON SPATIAL QUERY MX7
SELECT AVG(sum_perf) AS average_performance
FROM (SELECT parcel_id, SUM(performance) AS sum_perf
      FROM annual_performances
      WHERE parcel_id IN (SELECT parcel_id
                        FROM parcels
                        WHERE Left(parcel_id::text,3) = '100'
                        )
      AND plant_id = 'plant_15'
      GROUP BY parcel_id) AS T

-- NON SPATIAL QUERY MX8
-- DROP VIEW max_perf CASCADE;
CREATE VIEW max_perf AS
  SELECT parcel_id, SUM(performance) AS total_performance
  FROM annual_performances
  GROUP BY parcel_id
  ORDER BY SUM(performance) DESC
  LIMIT 5;
-- SELECT * FROM max_perf
```

```
SELECT parcel_id, COUNT(*)
FROM people
WHERE parcel_id IN (SELECT parcel_id
                    FROM max_perf)
GROUP BY parcel_id
```

5. Ο κώδικας SQL των χωρικών ερωτημάτων (αρχείο *'Spatial Queries.sql'*).

```
-- SPATIAL QUERY X1
SELECT ST_Area(
    (SELECT ST_UNION(
        ARRAY(
            SELECT parcels.geometry
            FROM parcels
            ORDER BY parcels.asmt DESC
            LIMIT 5
        )
    )
)

-- SPATIAL QUERY X2
SELECT ST_Perimeter(T.geometry)
FROM(
    SELECT parks.geometry
    FROM parks
    ORDER BY parks.size DESC
    LIMIT 1) AS T

-- SPATIAL QUERY X3
DROP TABLE query_layer;
SELECT parcels.*
INTO query_layer
FROM parcels, firm
WHERE ST_Contains(firm.geometry, parcels.geometry)
      AND firm.zone = 'X500'

-- SPATIAL QUERY X4
DROP TABLE query_layer;
SELECT parcels.*
INTO query_layer
FROM parcels, firm
WHERE ST_Intersects(firm.geometry, parcels.geometry)
      AND firm.zone = 'X500'

-- SPATIAL QUERY X5
DROP TABLE query_layer;
SELECT parcels.*
INTO query_layer
FROM parcels, firm
WHERE ST_Overlaps(firm.geometry, parcels.geometry)
```

```
AND firm.zone = 'X500'

-- SPATIAL QUERY X6
DROP TABLE query_layer;
SELECT parks.geometry
INTO query_layer
FROM parks, roads
WHERE ST_Crosses(parks.geometry,roads.geometry)

-- SPATIAL QUERY X7
SELECT park1_name, max(parks_dist) AS max_parks_dist
FROM (
  SELECT    p1.name    AS    park1_name,    p2.name    AS    park2_name,
  ST_Distance(p1.geometry, p2.geometry) AS parks_dist
  FROM parks AS p1, parks AS p2
  WHERE p1.name <> p2.name
  ORDER BY p1.name, parks_dist ASC
) AS PD
GROUP BY park1_name

-- SPATIAL QUERY X8
-- DROP VIEW Parks_Distances CASCADE;
CREATE VIEW Parks_Distances AS
  SELECT    p1.name    AS    park1_name,    p2.name    AS    park2_name,
  ST_Distance(p1.geometry, p2.geometry) AS parks_dist
  FROM parks AS p1, parks AS p2
  WHERE p1.name <> p2.name
  ORDER BY p1.name, parks_dist ASC;
-- SELECT * FROM Parks_Distances

-- DROP VIEW Min_Parks_Distance CASCADE;
CREATE VIEW Min_Parks_Distance AS
  SELECT park1_name, MIN(parks_dist) AS min_parks_dist
  FROM Parks_Distances
  GROUP BY park1_name;
-- SELECT * FROM Min_Parks_Distance

SELECT minpd.park1_name, pd.park2_name, minpd.min_parks_dist
FROM Min_Parks_Distance AS minpd, Parks_Distances AS pd
WHERE minpd.min_parks_dist = pd.parks_dist AND minpd.park1_name <>
pd.park2_name
ORDER BY minpd.min_parks_dist
```

8. ΒΙΒΛΙΟΓΡΑΦΙΑ – ΑΝΑΦΟΡΕΣ

1. Υλικό μαθήματος
2. [PostgreSQL 11.3 Documentation](#)
3. [PostGIS 2.5.3dev Manual](#)
4. [QGIS Documentation](#)
5. [Απεικόνιση Διαγράμματος Οντοτήτων-Συσχετίσεων σε Σχεσιακό Μοντέλο Δεδομένων](#)
6. [Spatial database – Wikipedia](#)
7. [Entity–relationship model – Wikipedia](#)
8. [Relational model – Wikipedia](#)