Συστήματα Διαχείρισης Βάσεων Δεδομένων 2018-19

(5ο εξάμηνο)

Ομάδα Εργασίας:p14086,Κουσουννής Κωνσταντίνος

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

Στοιχεία Φοιτητή

Επώνυμο Φοιτητή	Κουσουννής
Όνομα Φοιτητή	Κωνσταντίνος
Αριθμός Μητρώου	P14086
Email	Kwstas654321@gmail.com
Τιτλος	Εργασία Συστήματα Διαχείρισης Βάσεων Δεδομένων 2018-2019

Εργασία Συστήματα Διαχείρισης Βάσεων Δεδομένων 2018-2019

Εισαγωγή

Θα εργαστείτε με ένα σύνολο δεδομένων από την ιστοσελίδα ανοιχτών δεδομένων (Open Data) της κυβέρνησης του Ηνωμένου Βασιλείου, που έχουν δημοσιευθεί από το Υπουργείο Μεταφορών, συγκεκριμένα το: UK Road Safety: Traffic Accidents and Vehicles (URL: https://www.kaggle.com/tsiaras/uk-road-safety-accidents-andvehicles/home?fbclid=IwAR1fgYHabasR7ObkbySkApiImmDwo7LNs6PQ-7LEgxlpuRkU2x8OUVvIzc). Το συγκεκριμένο σύνολο δεδομένων αφορά σε αναλυτικές πληροφορίες για τροχαία ατυχήματα και εμπλεκόμενα οχήματα στο Ηνωμένο Βασίλειο κατά τα έτη 2005- 2016. Το υποσύνολο δεδομένων που θα χρησιμοποιήσετε και αποτελείται από δύο αρχεία csv, έχει υποστεί κατάλληλη επεξεργασία, ώστε να εξυπηρετεί τις ανάγκες και τους σκοπούς της εργασίας. Μπορείτε να το κατεβάσετε από τον ακόλουθο σύνδεσμο (URL: https://www.dropbox.com/s/lzxju6fnmcbi8fh/db2_data.zip?dl=0): • Accident_Information.csv: κάθε γραμμή του αρχείου αντιπροσωπεύει ένα μοναδικό τροχαίο ατύχημα (στήλη Accident_Index). • Vehicle_Information.csv: κάθε γραμμή του αρχείου αντιπροσωπεύει τη συμμετοχή ενός μοναδικού οχήματος σε ένα μοναδικό τροχαίο ατύχημα. Βάσει των παραπάνω, καλείστε να απαντήσετε στα ακόλουθα ερωτήματα:

Ερώτημα 1 (60%). Βελτιστοποίηση Επερωτήσεων σε Κεντρικοποιημένη ΒΔ

Κάθε φορά που εκτελείτε μία από τις παρακάτω επερωτήσεις θα δείχνετε τον χρόνο εκτέλεσης (πάντα θα εκτελείτε την επερώτηση τουλάχιστον δύο φορές και θα κρατάτε τον τελευταίο χρόνο – ο λόγος που δεν κρατάμε απλά τον χρόνο εκτέλεσης της πρώτης φοράς είναι ότι δεν είναι αντιπροσωπευτικός γιατί τα buffers δεν έχουν προλάβει να αρχικοποιηθούν) καθώς και το πλάνο εκτέλεσης (χρησιμοποιώντας την εντολή EXPLAIN, screenshot). Σκοπός είναι κάθε φορά που αλλάζετε κάτι στη ΒΔ, με απώτερο στόχο να βελτιώσετε τους χρόνους εκτέλεσης, να παρατηρείτε αν υπάρχει βελτίωση και πόση είναι αυτή αλλά και να εξηγείτε τη βελτίωση αυτή με βάση τη θεωρία και το πλάνο εκτέλεσης. Η απάντηση στα ερωτήματα a, b, c, d, e πρέπει να γίνει με τη σειρά εμφάνισής τους, δηλαδή, θα απαντήσετε στο ερώτημα b αφού πρώτα έχετε απαντήσει στο a (και ούτω καθεξής), έτσι ώστε οι αλλαγές που κάνατε στο a (π.χ. buffers, parallelism, κτλ.) να συνεχίσουν να είναι ενεργές στο b και έπειτα

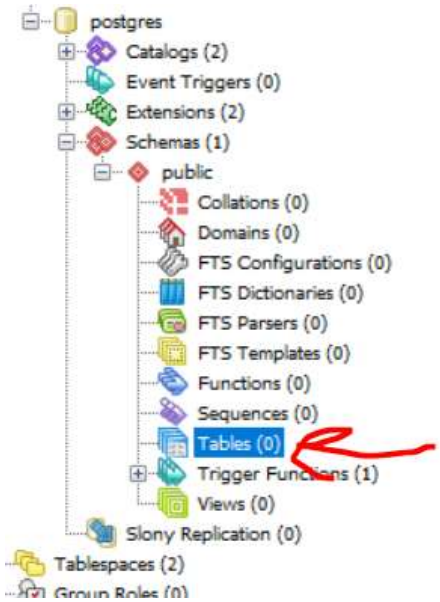
(a) Αφού φορτώσετε τα δεδομένα στο Σύστημα Διαχείρισης ΒΔ της PostgreSQL (εντολή "COPY … WITH CSV HEADER") και ανανεώσετε τα στατιστικά χρησιμοποιώντας την εντολή "VACUUM FULL …", εκτελέστε τις παρακάτω επερωτήσεις (queries) χρησιμοποιώντας τις

προεπιλεγμένες ρυθμίσεις της PostgreSQL και χωρίς να έχετε δημιουργήσει βοηθητικές δομές (π.χ. ευρετήρια).

Αρχικά κατεβάζουμε τα δεδομένα μας από την ιστοσελίδα που μας δίνεται.

Τα δύο αρχεία ονομάζονται **db2_Accident_information,db2_Vehicle_Information**

Αφού κατεβάσουμε τα αρχεία μας ανοίγουμε το pgadmin3 και δημιουργούμε δυο πίνακες με τα αντίστοιχα χαρακτηριστικά.



Ανοίγουμε ένα query και δίνουμε τις εντολές μας.

Δηλώνουμε τον πίνακα db2_Accident_Information

```
create table db2_Accident_Information(
empty integer,
Accident_Index varchar(255),
First_Road_Class varchar(255),
Accident_Severity varchar(255),
Date date,
Urban_or_Rural_Area varchar(255),
Weather_Conditions varchar(255),
Year integer,
InScotland varchar(255),
primary key(Accident_Index)
);

copy db2_Accident_Information from 'D:\db2_Accident_Information.csv' Delimiter ',' CSV HEADER;
```

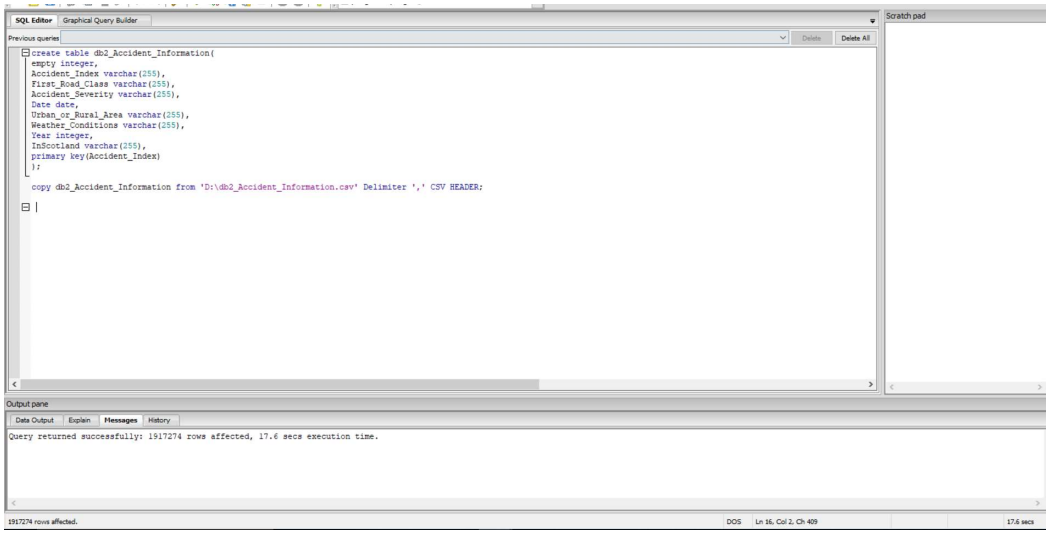
Αυτός έχει τις αντιστοιχες στήλες

	A	B	C	D	E	F	G	H	I	J
	Accident_Index	1st_Road_Class	Accident_Severity	Date	Urban_or_Rural_Ar	Weather_Conditions	Year	InScotland		
0	200501BS00001	A	Serious	1/4/2005	Urban	Raining no high winds	2005	No		

Κάθε στήλη που έχω ορίσει είναι η αντίστοιχη με την στήλη που έχω μέσα στο csv αρχείο όπως βλέπουμε και παραπάνω.

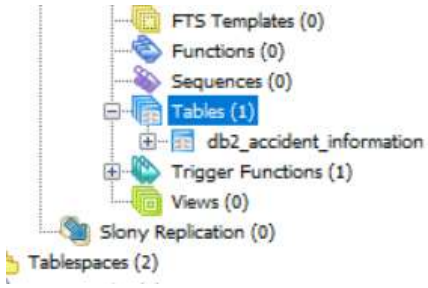
Κα βάζουμε σαν πρωτεύων κλειδί (**primary key**) την στήλη Accident_Index η οποία δεν έχει κανένα ίδιο στοιχείο σύμφωνα με την εκφώνηση της άσκησης.

- Accident_Information.csv: κάθε γραμμή του αρχείου αντιπροσωπεύει ένα μοναδικό τροχαίο ατύχημα (στήλη Accident_Index).



Τα αρχεία μου τα έχω αποθηκευμένα στο path 'D:\db2_Accident_Information'

Τρέχουμε το query και βλέπουμε ότι τελείωσε επιτυχημένα.



Πηγαίνουμε μετά στην βάση μας και ελέγχουμε ναν έχουν περαστεί η πίνακες.

Οπως βλέπουμε ο πίνακας έχει δημιουργηθεί κοιτάμε και τα δεδομένα.

File Edit View Tools Help

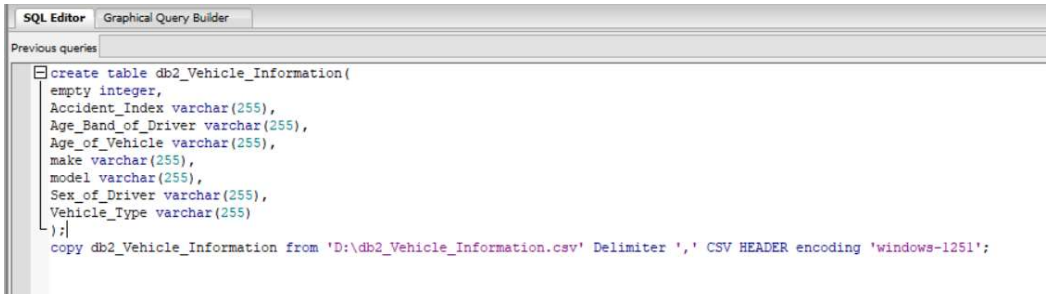
100 rows

	empty integer	accident_index [PK] character varying(255)	first_road_class character varying(255)	accident_severity character varying(255)	date date	urban_or_rural_area character varying(255)	weather_conditions character varying(255)	year integer	inscotland character varying(255)
1	0	200501BS00001	A	Serious	2005-01-04	Urban	Raining no high winds	2005	No
2	1	200501BS00002	B	Slight	2005-01-05	Urban	Fine no high winds	2005	No
3	2	200501BS00003	C	Slight	2005-01-06	Urban	Fine no high winds	2005	No
4	3	200501BS00004	A	Slight	2005-01-07	Urban	Fine no high winds	2005	No
5	4	200501BS00005	Unclassified	Slight	2005-01-10	Urban	Fine no high winds	2005	No
6	5	200501BS00006	Unclassified	Slight	2005-01-11	Urban	Raining no high winds	2005	No
7	6	200501BS00007	C	Slight	2005-01-13	Urban	Fine no high winds	2005	No
8	7	200501BS00009	A	Slight	2005-01-14	Urban	Fine no high winds	2005	No
9	8	200501BS00010	A	Slight	2005-01-15	Urban	Fine no high winds	2005	No
10	9	200501BS00011	B	Slight	2005-01-15	Urban	Fine no high winds	2005	No
11	10	200501BS00012	A	Slight	2005-01-16	Urban	Fine no high winds	2005	No
12	11	200501BS00014	A	Slight	2005-01-25	Urban	Fine no high winds	2005	No
13	12	200501BS00015	Unclassified	Slight	2005-01-11	Urban	Raining no high winds	2005	No
14	13	200501BS00016	A	Slight	2005-01-18	Urban	Raining no high winds	2005	No
15	14	200501BS00017	A	Slight	2005-01-18	Urban	Fine no high winds	2005	No
16	15	200501BS00018	A	Slight	2005-01-18	Urban	Fine no high winds	2005	No
17	16	200501BS00019	Unclassified	Serious	2005-01-20	Urban	Fine no high winds	2005	No
18	17	200501BS00020	A	Slight	2005-01-21	Urban	Fine no high winds	2005	No
19	18	200501BS00021	B	Slight	2005-01-21	Urban	Fine no high winds	2005	No
20	19	200501BS00022	A	Serious	2005-01-08	Urban	Fine no high winds	2005	No
21	20	200501BS00023	Unclassified	Slight	2005-01-24	Urban	Fine no high winds	2005	No
22	21	200501BS00024	B	Slight	2005-01-24	Urban	Fine no high winds	2005	No
23	22	200501BS00025	A	Slight	2005-01-24	Urban	Fine no high winds	2005	No
24	23	200501BS00028	C	Slight	2005-01-18	Urban	Fine no high winds	2005	No
25	24	200501BS00029	A	Slight	2005-01-29	Urban	Fine no high winds	2005	No
26	25	200501BS00031	C	Slight	2005-01-19	Urban	Raining no high winds	2005	No

Scratch pad

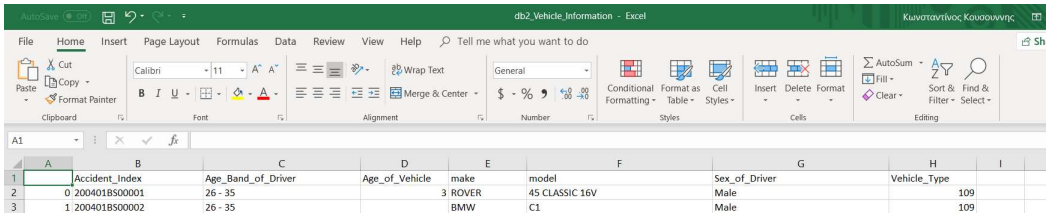
Όπως βλέπουμε τα δεδομένα μας έχουν περάσει κανονικά.

Ακολουθούμε την ίδια διαδικασία για τον επόμενο πίνακα.



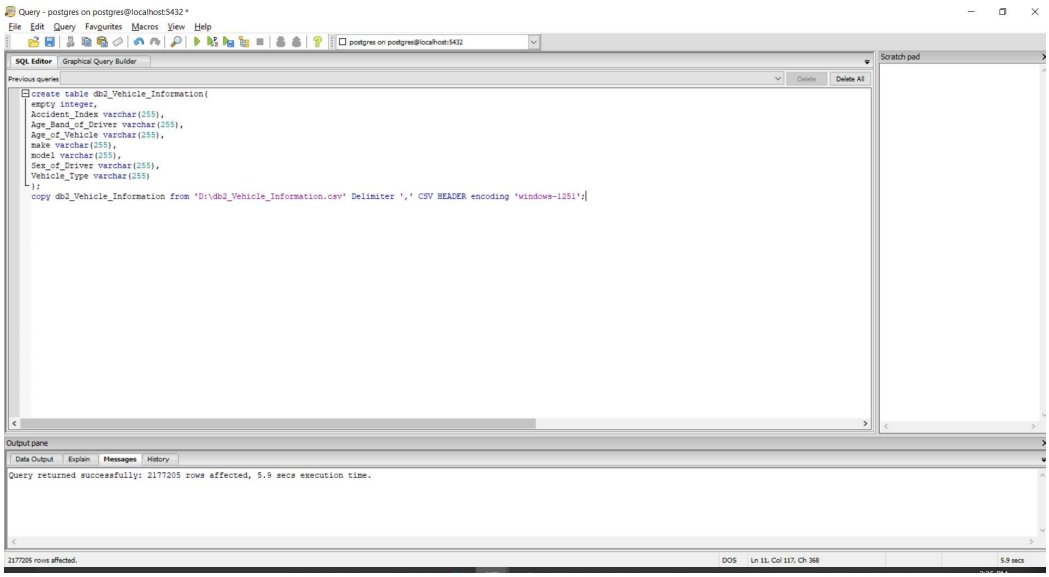
Δημιουργούμε έναν πίνακα db2_Vehicle_Information

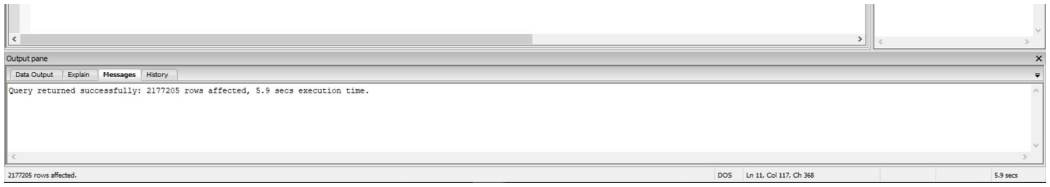
δηλώνουμε τις στήλες μας αντίστοιχα με το db2_Vehicle_Information.csv



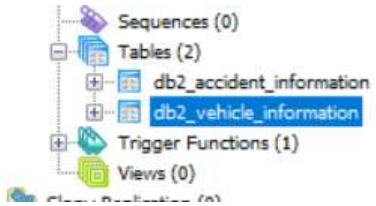
Εχω αντίστοιχα το αρχείο αποθηκευμένο στο path ‘D:\db2_Vehicle_Information’

Πατάω play και περνάω τα στοιχεία μου.





Τα δεδομένα μου πέρασαν με επιτυχία.



Ο πίνακας έχει οριστεί.

Edit Data - PostgreSQL 9.5 (localhost:5432) - postgres - public.db2_vehicle_information

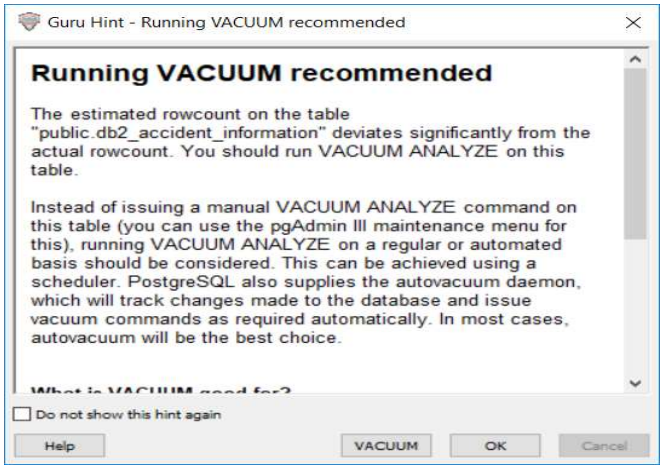
File Edit View Tools Help

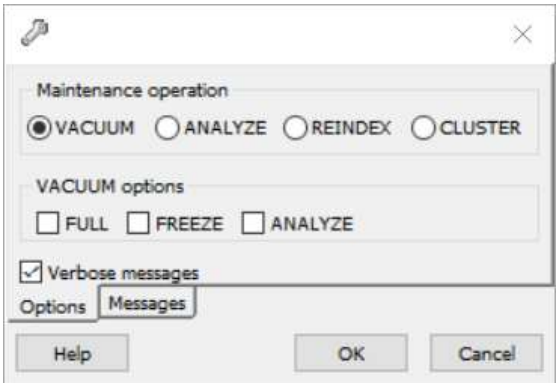
100 rows

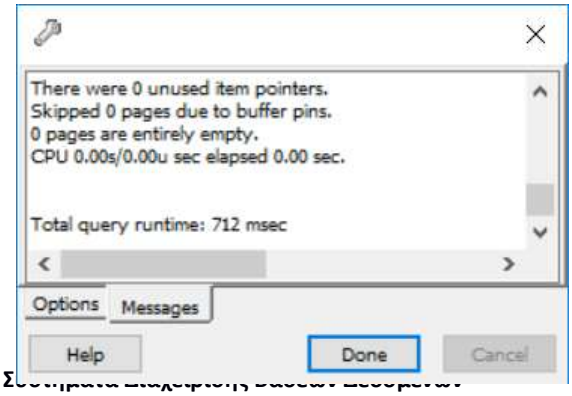
	empty integer	accident_index character varying(255)	age_band_of_driver character varying(255)	age_of_vehicle character varying(255)	make character varying(255)	model character varying(255)	sex_of_driver character varying(255)	vehicle_type character varying(255)
1	0	200401BS00001	26 - 35	3.0	ROVER	45 CLASSIC 16V	Male	109
2	1	200401BS00002	26 - 35		BMW	C1	Male	109
3	2	200401BS00003	26 - 35	4.0	NISSAN	MICRA CELEBRATION 16V	Male	109
4	3	200401BS00003	66 - 75		LONDON TAXIS INT	TXII GOLD AUTO	Male	109
5	4	200401BS00004	26 - 35	1.0	PIAGGIO	VESPA ET4	Male	Motorcycle 125cc and under
6	5	200401BS00004	36 - 45	10.0	VOLKSWAGEN		Male	109
7	6	200401BS00009	26 - 35		PIAGGIO	VESPA GT 125	Male	Motorcycle 125cc and under
8	7	200401BS00010	36 - 45		BMW	R1100 RT	Male	109
9	8	200401BS00012	46 - 55	3.0	MERCEDES		Male	109
10	9	200401BS00013	26 - 35	4.0	VOLKSWAGEN	GOLF V5	Female	109

Κάνουμε Vaccum και στους δυο πίνακες

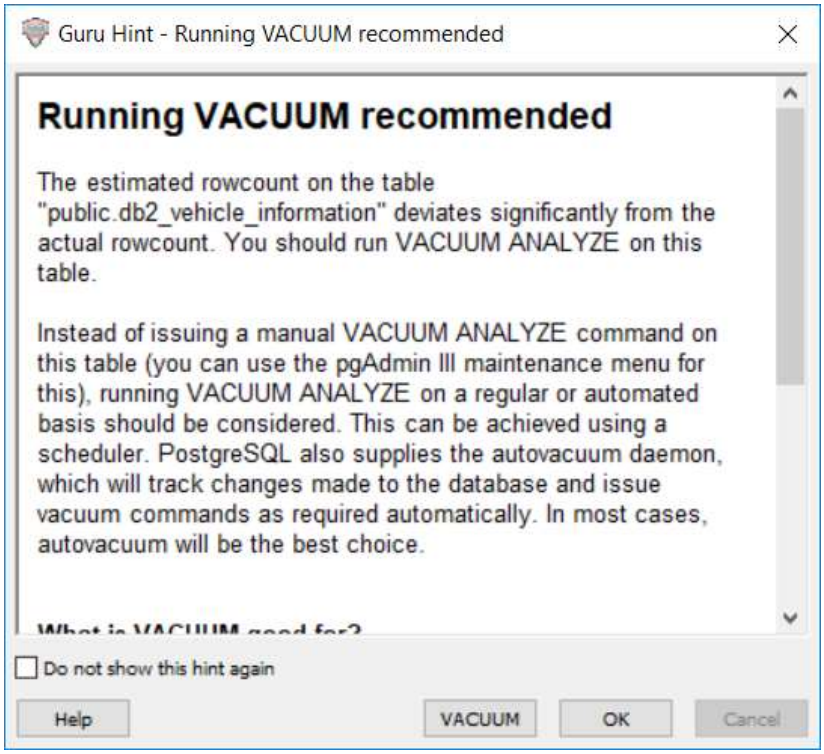
db2_accident_information

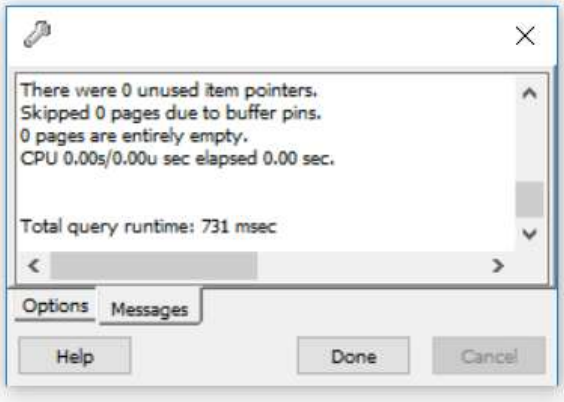


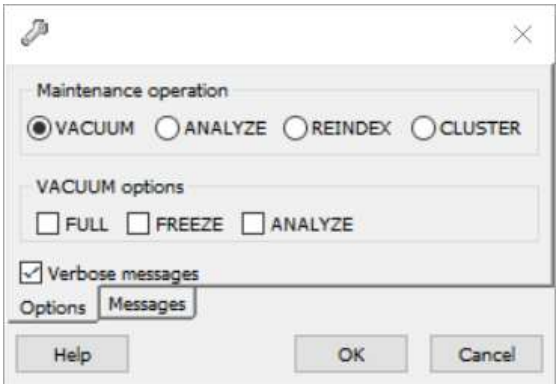


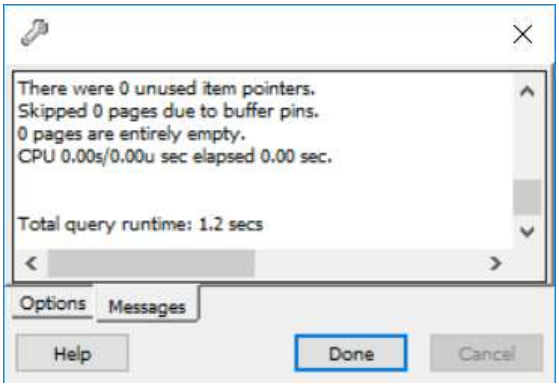


db2_vehicle_information





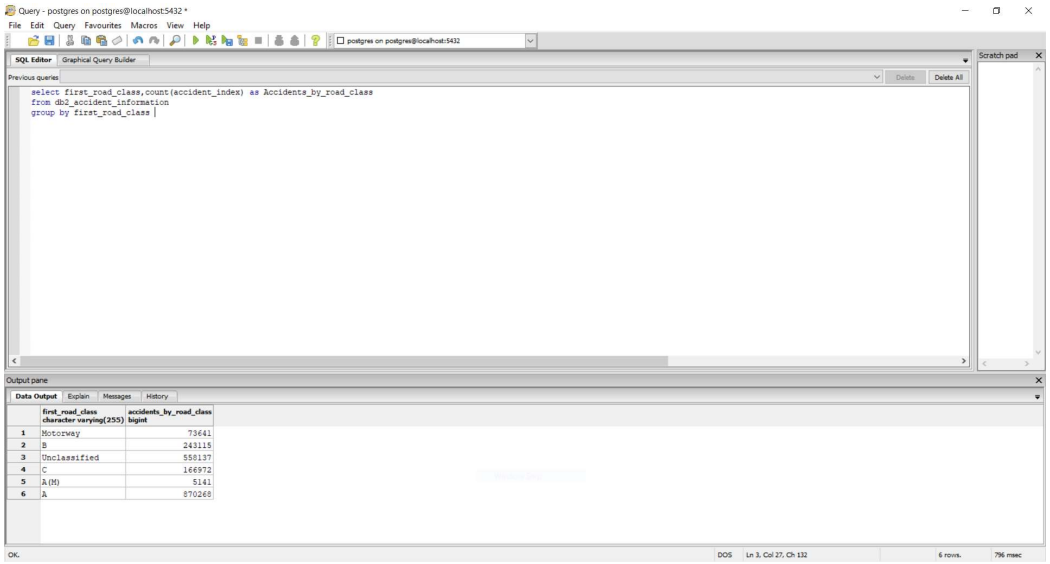


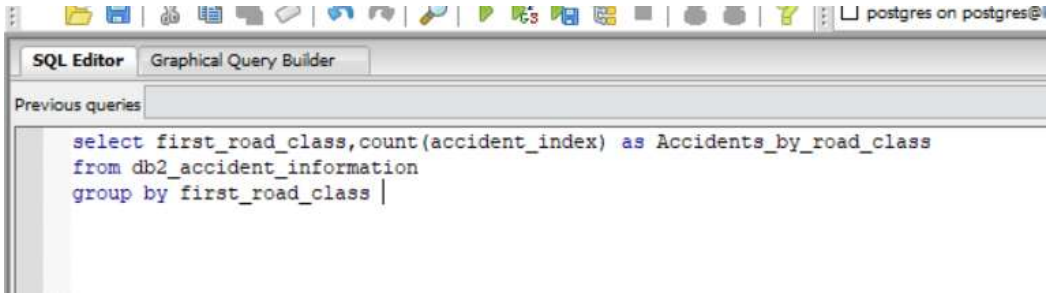


Σε κάθε εκτέλεση κάθε ερωτήματος έχω μια **1η εκτέλεση** και μια **2η εκτέλεση εμείς θα παίρνουμε και μας ενδιαφέρει η 2η εκτέλεση(αν και λέω δεύτερη εκτέλεση μπορεί να έχω τρέξει περισσότερες από τρείς φορές την επερώτηση) από την εκφώνηση:**

(πάντα θα εκτελείτε την επερώτηση τουλάχιστον δύο φορές και θα κρατάτε τον τελευταίο χρόνο – ο λόγος που δεν κρατάμε απλά τον χρόνο εκτέλεσης της πρώτης φοράς είναι ότι δεν είναι αντιπροσωπευτικός γιατί τα buffers δεν έχουν προλάβει να αρχικοποιηθούν.

i. Βρείτε πόσα ατυχήματα έγιναν ανά κατηγορία δρόμου (road class).


Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
SQL Editor
Graphical Query Builder
Previous queries
Scratch pad
Delete
Delete All
select first_road_class,count(accident_index) as Accidents_by_road_class
from db2_accident_information
group by first_road_class
Output pane
Data Output
Explain
Messages
History
first_road_class character varying(255)
accidents_by_road_class bigint
1 Motorway 73641
2 B 243115
3 Unclassified 558137
4 C 166972
5 A(M) 5141
6 A 870268
OK.
DOS
Ln 3, Col 27, Ch 132
6 rows.
796 msec


postgres on postgres@
SQL Editor
Graphical Query Builder
Previous queries
select first_road_class,count(accident_index) as Accidents_by_road_class
from db2_accident_information
group by first_road_class

Θέλουμε να μας εμφανίσει τα ατυχήματα που έγιναν ανά κατηγορία δρόμου.

select first_road_class,count(accident_index) as Accidents_by_road_class

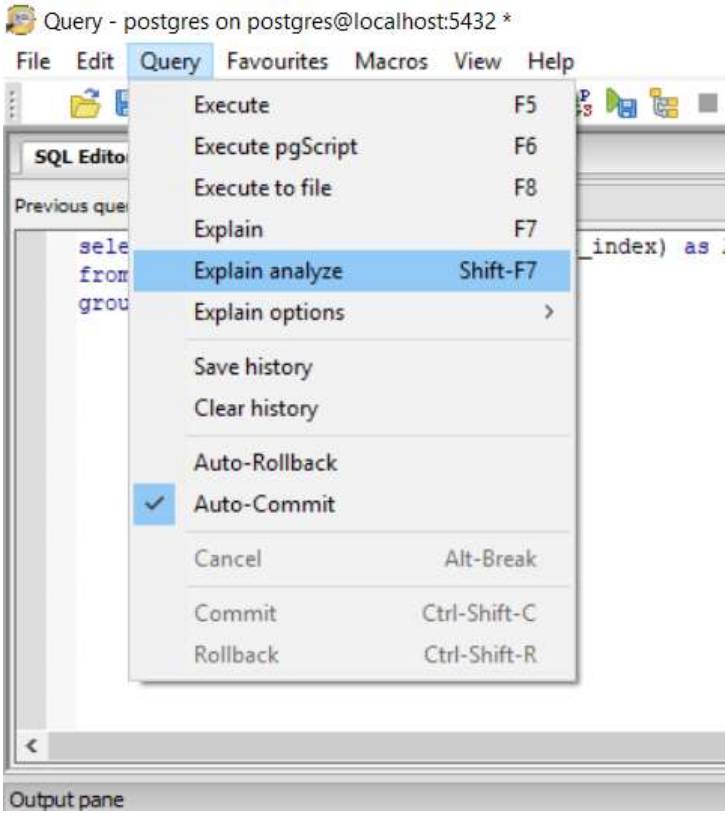
from db2_accident_information

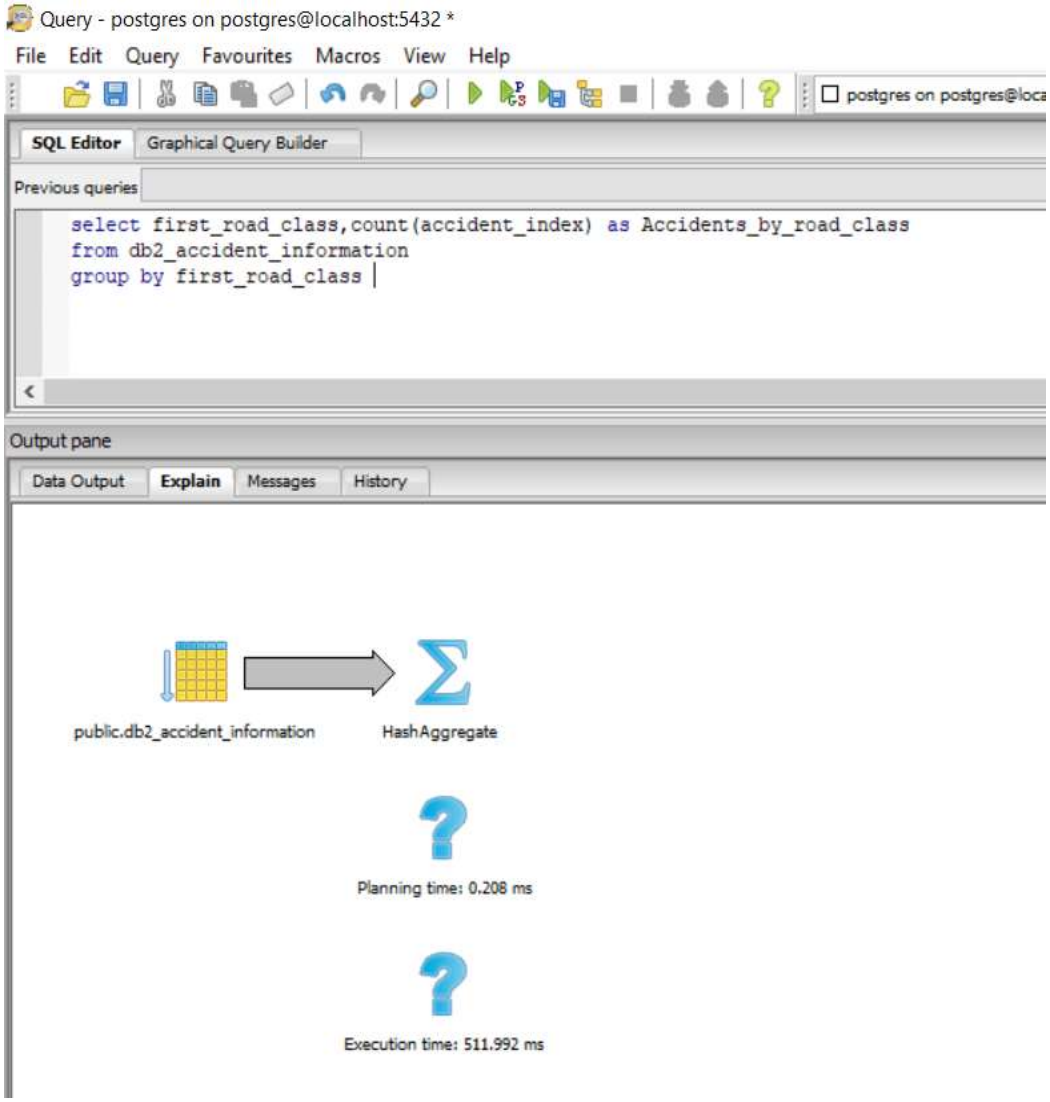
group by first_road_class

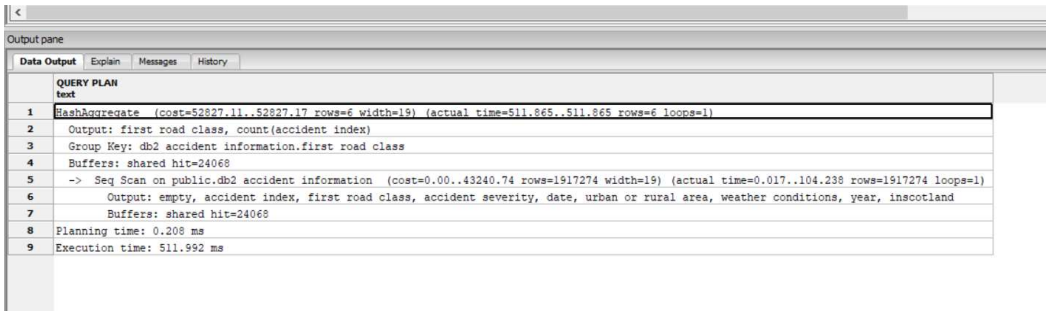
οπότε με την count μετράμε ποσά είναι τα ατυχήματα αφού τα κάνουμε ομαδοποίηση με βάση τον δρόμο

Στην συνέχεια πατάμε explain query

1η Εκτέλεση




Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
SQL Editor
Graphical Query Builder
Previous queries
select first_road_class,count(accident_index) as Accidents_by_road_class
from db2_accident_information
group by first_road_class
Output pane
Data Output
Explain
Messages
History
public.db2_accident_information
HashAggregate
Planning time: 0.208 ms
Execution time: 511.992 ms



Output pane

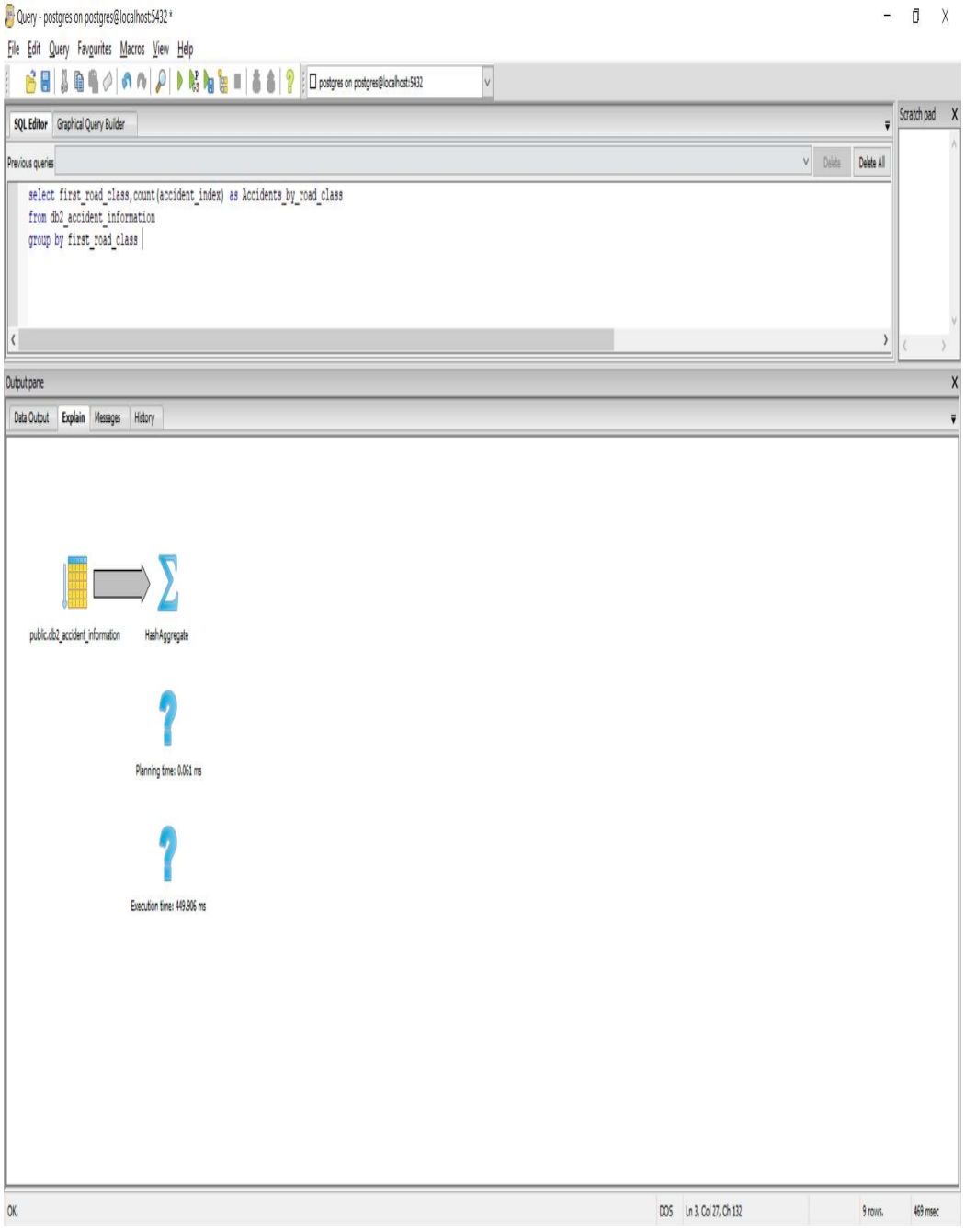
Data Output | Explain | Messages | History

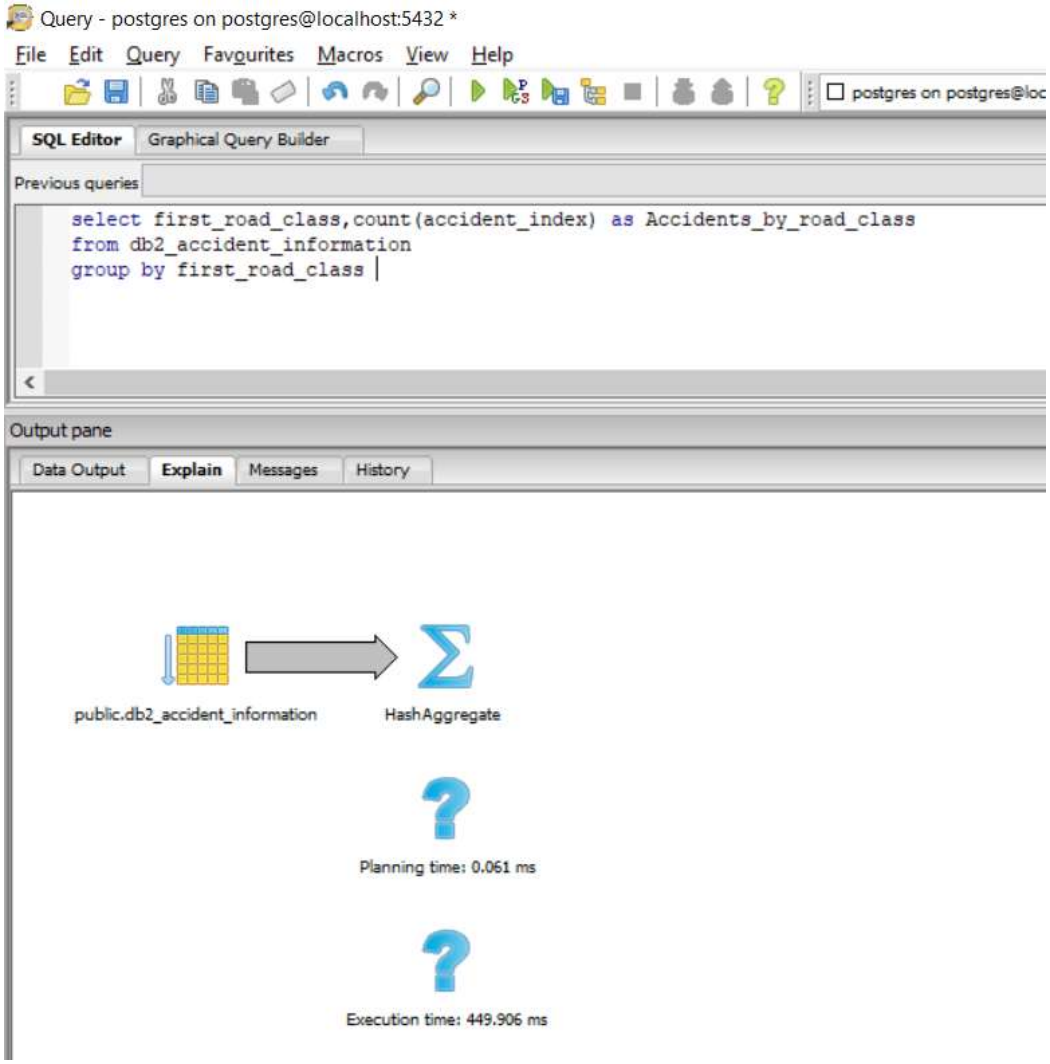
	QUERY PLAN text
1	HashAggregate (cost=52827.11..52827.17 rows=6 width=19) (actual time=511.865..511.865 rows=6 loops=1)
2	Output: first road class, count(accident index)
3	Group Key: db2 accident information.first road class
4	Buffers: shared hit=24068
5	-> Seq Scan on public.db2 accident information (cost=0.00..43240.74 rows=1917274 width=19) (actual time=0.017..104.238 rows=1917274 loops=1)
6	Output: empty, accident index, first road class, accident severity, date, urban or rural area, weather conditions, year, inscotland
7	Buffers: shared hit=24068
8	Planning time: 0.208 ms
9	Execution time: 511.992 ms

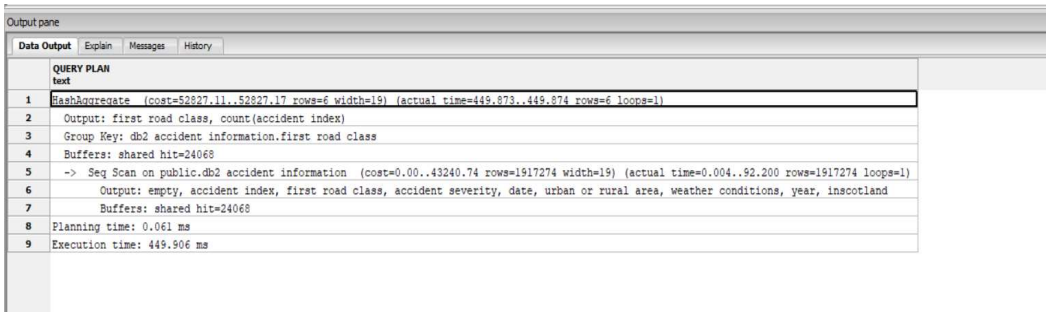
Planning time:0.208ms

Execution time:511.992ms

2η Εκτέλεση(αυτή που μας ενδιαφέρει και κρατάμε(αν και λέω δεύτερη εκτέλεση μπορεί να έχω τρέξει περισσότερες από τρείς φορές την επερώτηση))


Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
postgres on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Scratch pad
Previous queries
Delete
Delete All
select first_road_class,count(accident_index) as Accidents_by_road_class
from db2_accident_information
group by first_road_class
Output pane
Data Output
Explain
Messages
History
public.db2_accident_information
Hash Aggregate
Planning time: 0.061 ms
Execution time: 449.906 ms
OK.
DOS
Ln 3, Col 27, Ch 132
9 rows.
469 msec


Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
postgres on postgres@loc
SQL Editor
Graphical Query Builder
Previous queries
select first_road_class,count(accident_index) as Accidents_by_road_class
from db2_accident_information
group by first_road_class
Output pane
Data Output
Explain
Messages
History
public.db2_accident_information
HashAggregate
Planning time: 0.061 ms
Execution time: 449.906 ms


Output pane

Data Output | Explain | Messages | History

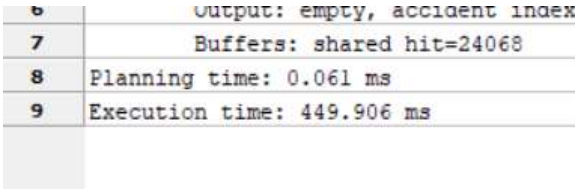
	QUERY PLAN text
1	HashAggregate (cost=52827.11..52827.17 rows=6 width=19) (actual time=449.873..449.874 rows=6 loops=1)
2	Output: first road class, count(accident index)
3	Group Key: db2 accident information.first road class
4	Buffers: shared hit=24068
5	-> Seq Scan on public.db2 accident information (cost=0.00..43240.74 rows=1917274 width=19) (actual time=0.004..92.200 rows=1917274 loops=1)
6	Output: empty, accident index, first road class, accident severity, date, urban or rural area, weather conditions, year, inscotland
7	Buffers: shared hit=24068
8	Planning time: 0.061 ms
9	Execution time: 449.906 ms

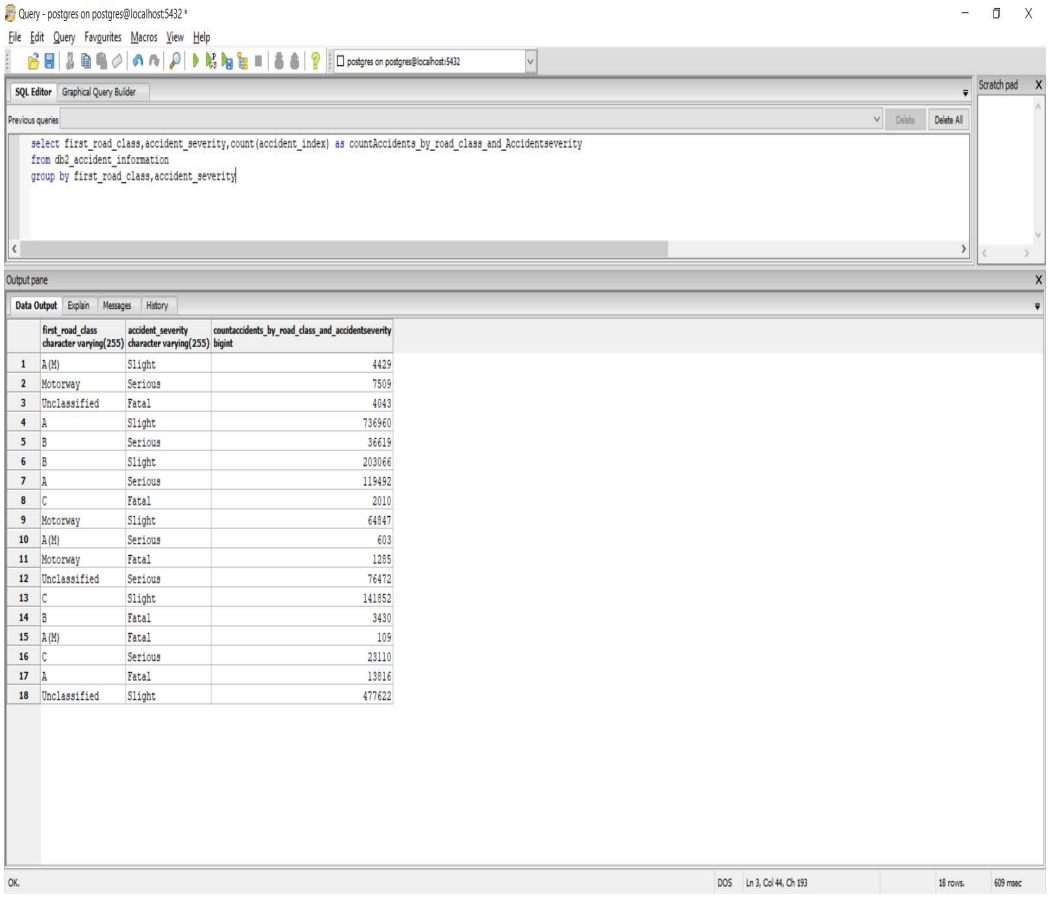
Planning time:0.061ms

Execution time:449.906ms

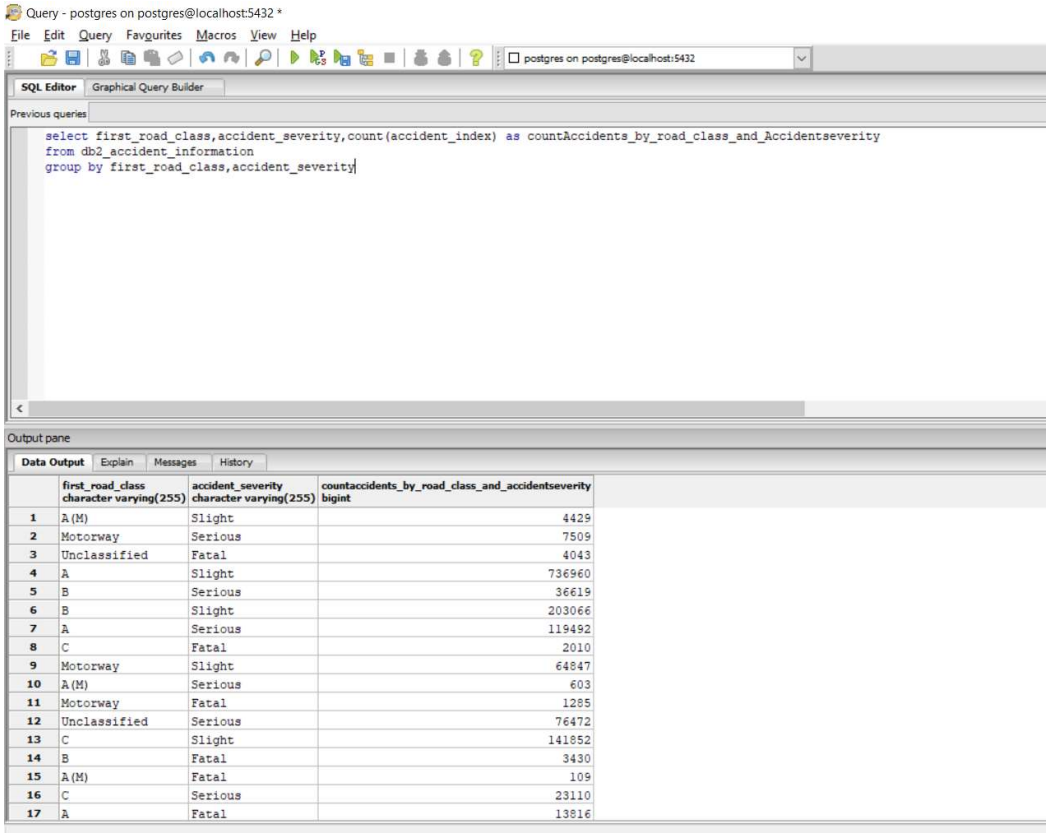
time:469msecs

ii. Πόσα είναι τα ατυχήματα ανά κατηγορία δρόμου και ανά κατηγορία ατυχήματος (Accident Severity).

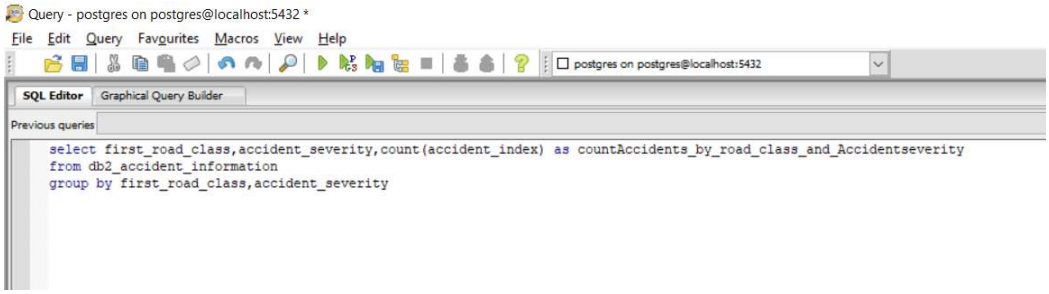




	first_road_class character varying(255)	accident_severity character varying(255)	countaccidents_by_road_class_and_accidentseverity bigint
1	A (M)	Slight	4429
2	Motorway	Serious	7509
3	Unclassified	Fatal	4043
4	A	Slight	736960
5	B	Serious	36619
6	B	Slight	203066
7	A	Serious	119492
8	C	Fatal	2010
9	Motorway	Slight	64847
10	A (M)	Serious	603
11	Motorway	Fatal	1285
12	Unclassified	Serious	76472
13	C	Slight	141852
14	B	Fatal	3430
15	A (M)	Fatal	109
16	C	Serious	23110
17	A	Fatal	13816
18	Unclassified	Slight	477622

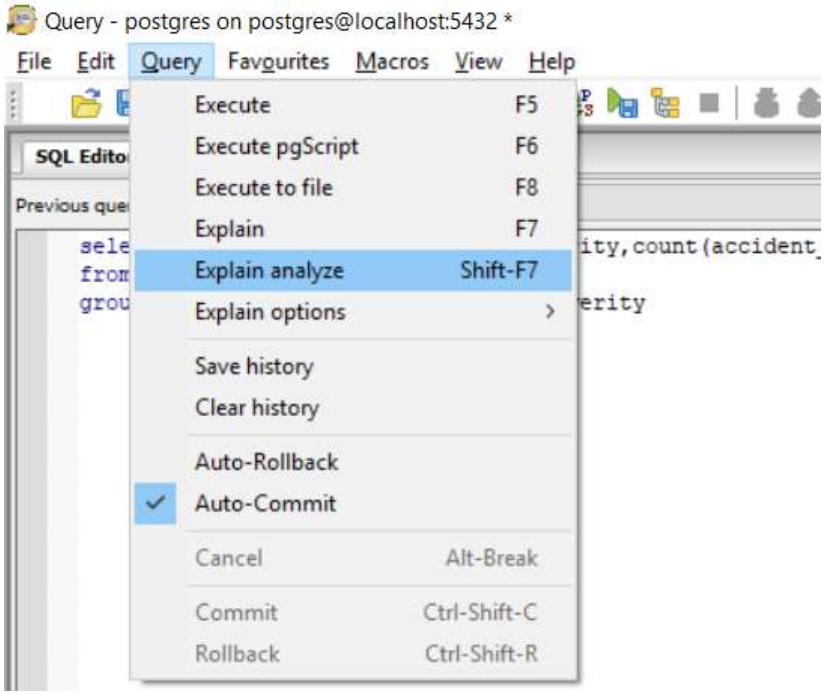


	first_road_class character varying(255)	accident_severity character varying(255)	countaccidents_by_road_class_and_accidentseverity bigint
1	A(M)	Slight	4429
2	Motorway	Serious	7509
3	Unclassified	Fatal	4043
4	A	Slight	736960
5	B	Serious	36619
6	B	Slight	203066
7	A	Serious	119492
8	C	Fatal	2010
9	Motorway	Slight	64847
10	A(M)	Serious	603
11	Motorway	Fatal	1285
12	Unclassified	Serious	76472
13	C	Slight	141852
14	B	Fatal	3430
15	A(M)	Fatal	109
16	C	Serious	23110
17	A	Fatal	13816

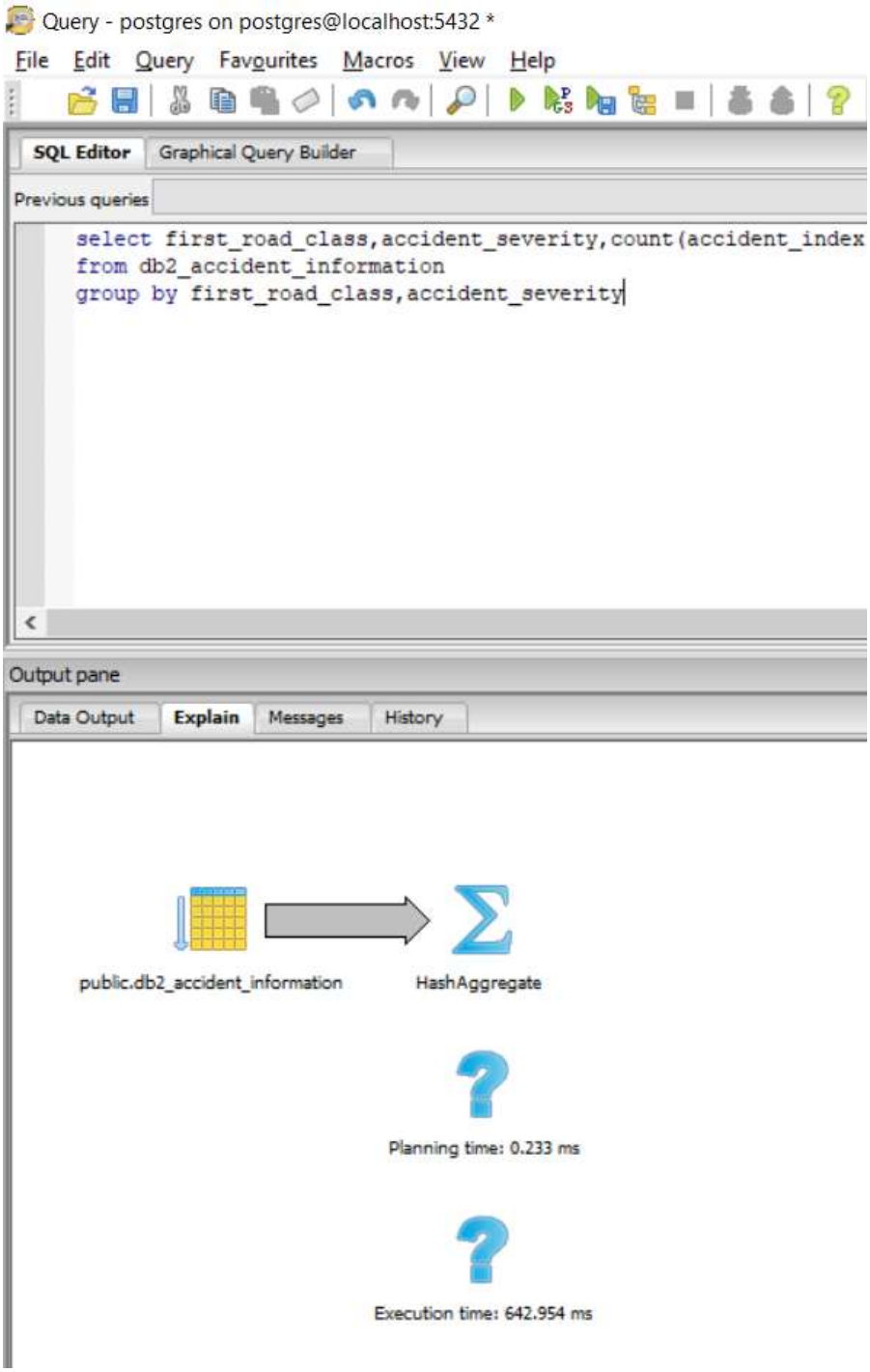


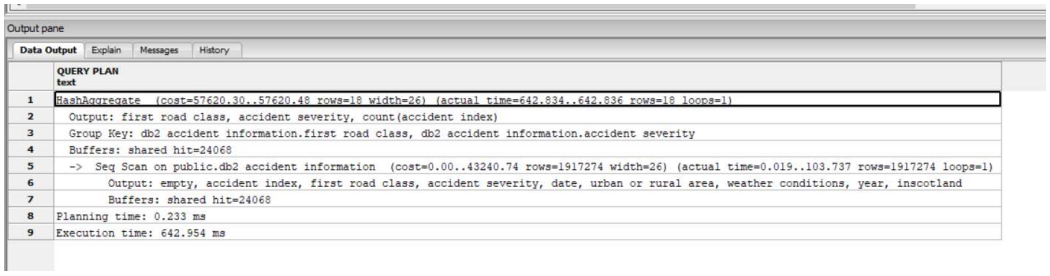
Θέλω να βρω πόσα είναι τα ατυχήματα ανά κατηγορία δρόμου και ανά κατηγορία ατυχήματος (Accident Severity).

Κάνω select για να εμφανίσω τον δρόμο την σοβαρότητα του ατυχήματος και πόσα είναι αυτά τα ατυχήματα και τα ομαδοποιώ με βάση τον δρόμο και την σοβαρότητα.



1η εκτέλεση

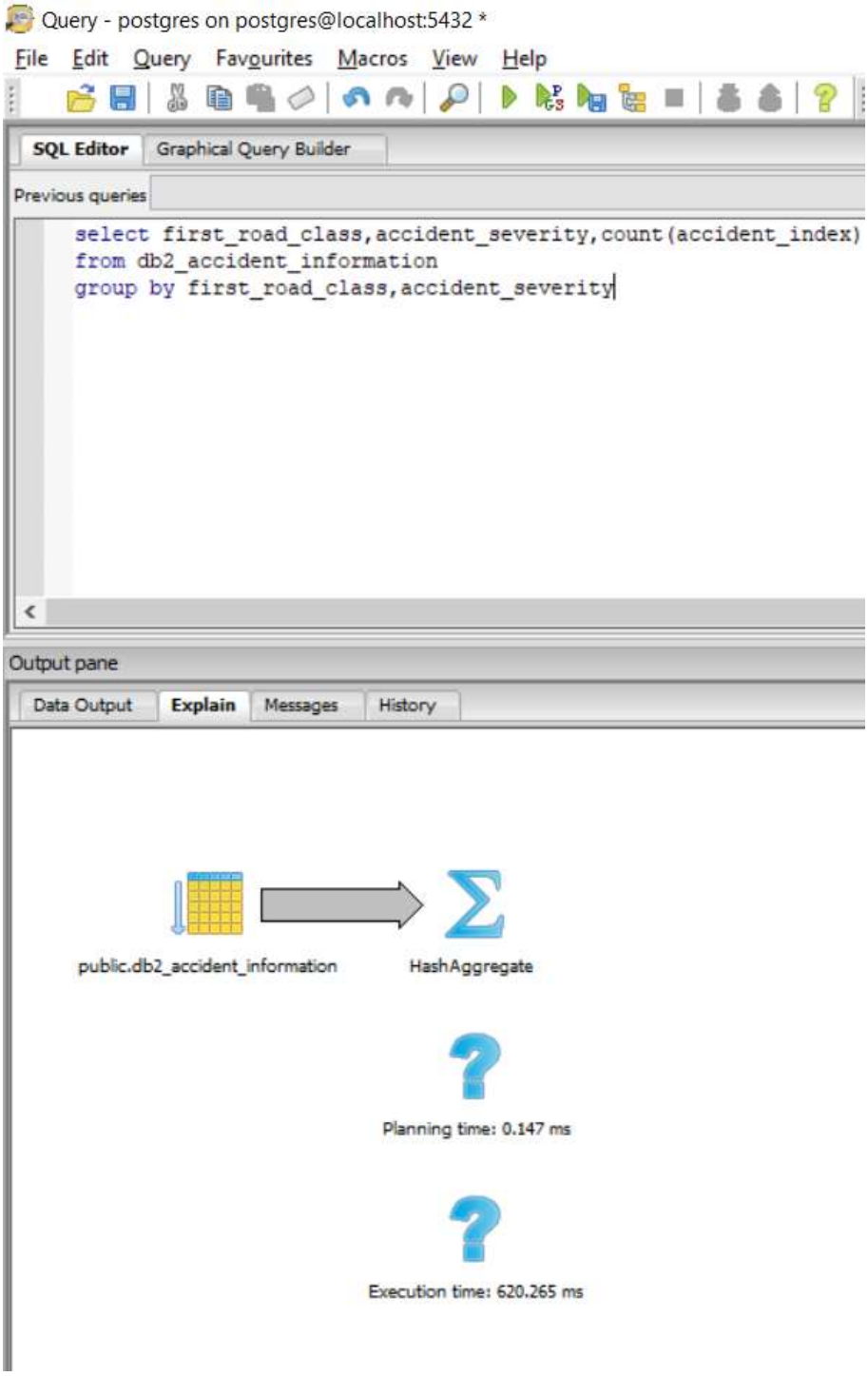

Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
SQL Editor
Graphical Query Builder
Previous queries
select first_road_class,accident_severity,count(accident_index
from db2_accident_information
group by first_road_class,accident_severity
Output pane
Data Output
Explain
Messages
History
public.db2_accident_information
HashAggregate
Planning time: 0.233 ms
Execution time: 642.954 ms

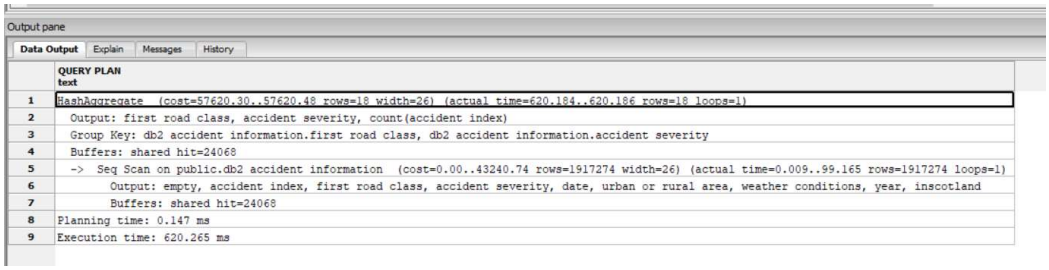

Output pane

Data Output | Explain | Messages | History

	QUERY PLAN text
1	HashAggregate (cost=57620.30..57620.48 rows=18 width=26) (actual time=642.834..642.836 rows=18 loops=1)
2	Output: first road class, accident severity, count(accident index)
3	Group Key: db2 accident information.first road class, db2 accident information.accident severity
4	Buffers: shared hit=24068
5	-> Seq Scan on public.db2 accident information (cost=0.00..43240.74 rows=1917274 width=26) (actual time=0.019..103.737 rows=1917274 loops=1)
6	Output: empty, accident index, first road class, accident severity, date, urban or rural area, weather conditions, year, inscotland
7	Buffers: shared hit=24068
8	Planning time: 0.233 ms
9	Execution time: 642.954 ms

2η Εκτέλεση(αυτή που μας ενδιαφέρει και κρατάμε(αν και λέω δεύτερη εκτέλεση μπορεί να έχω τρέξει περισσότερες από τρείς φορές την επερώτηση))


Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
SQL Editor
Graphical Query Builder
Previous queries
select first_road_class,accident_severity,count(accident_index)
from db2_accident_information
group by first_road_class,accident_severity
Output pane
Data Output
Explain
Messages
History
public.db2_accident_information
HashAggregate
Planning time: 0.147 ms
Execution time: 620.265 ms

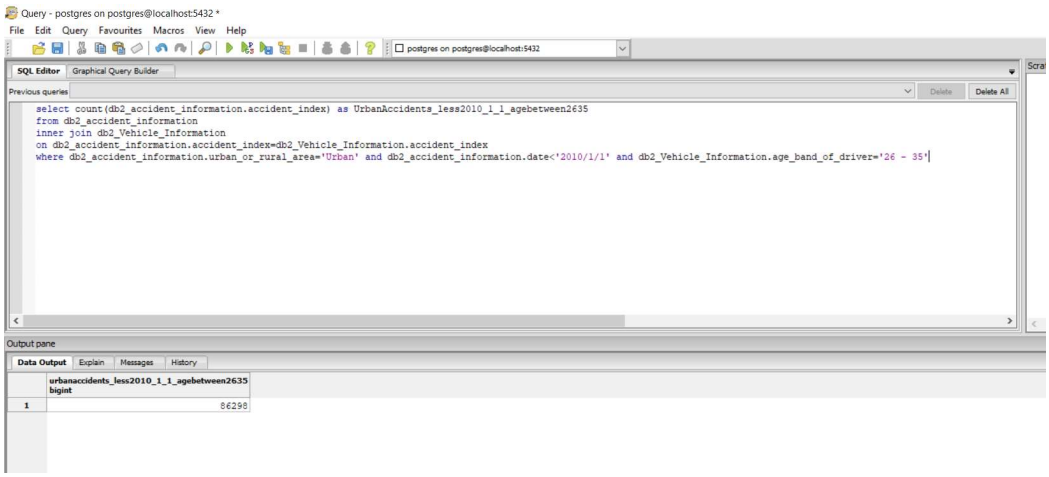


Planning time:0.147ms

Execution time:620.265ms

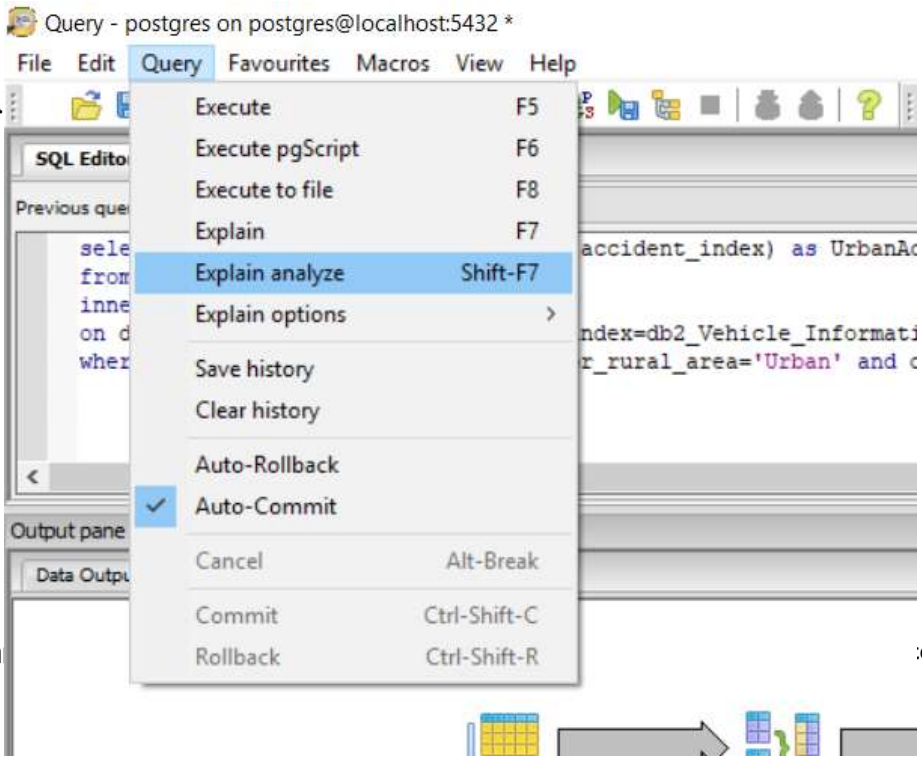
time:641msecs

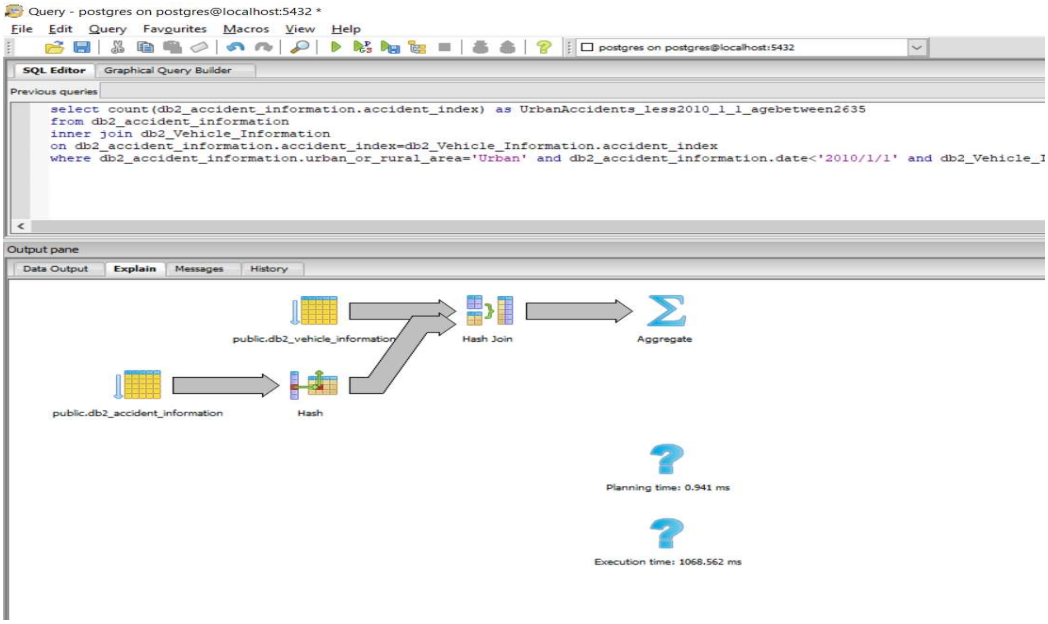
iii. Βρείτε πόσα ατυχήματα έγιναν σε αστική περιοχή πριν από την 1/1/2010 και η ηλικιακή κατηγορία του οδηγού είναι 26-35.



Επιλέγουμε πόσα ατυχήματα έγιναν σε αστική περιοχή οπότε κάνουμε count την στήλη accident index αλλά επειδή θέλουμε να είναι σε αστική περιοχή πριν από την 1/1/2010 και η ηλικιακή κατηγορία του οδηγού είναι 26-35 ενώνουμε τους δυο πίνακες με inner join για να κάνουμε ελέγχους από τους πίνακες db2_accident_information,db2_Vehicle_information και τέλος βάζουμε την συνθήκη για τους περιορισμούς μας **where db2_accident_information.urban_or_rural_area='Urban' and db2_accident_information.date<'2010/1/1' and db2_Vehicle_Information.age_band_of_driver='26 – 35'**

1η εκτέλεση




Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
postgres on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
select count(db2_accident_information.accident_index) as UrbanAccidents_less2010_1_1_agebetween2635
from db2_accident_information
inner join db2_Vehicle_Information
on db2_accident_information.accident_index=db2_Vehicle_Information.accident_index
where db2_accident_information.urban_or_rural_area='Urban' and db2_accident_information.date<'2010/1/1' and db2_Vehicle_I
Output pane
Data Output
Explain
Messages
History
public.db2_vehicle_information
Hash Join
Aggregate
public.db2_accident_information
Hash
Planning time: 0.941 ms
Execution time: 1068.562 ms

Output pane

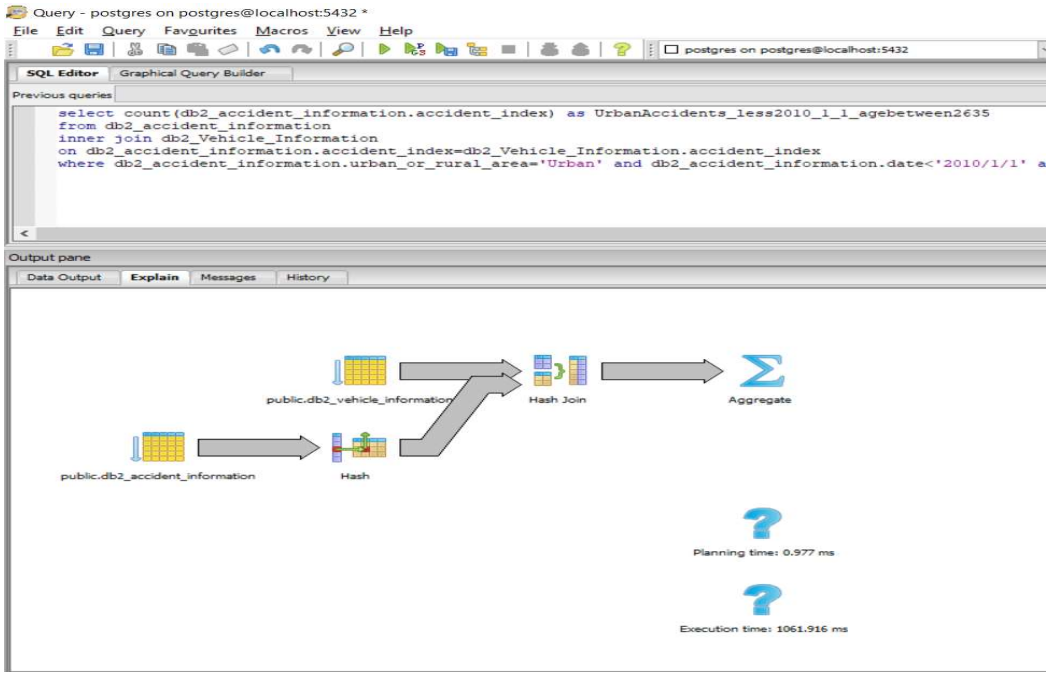
Data Output | Explain | Messages | History

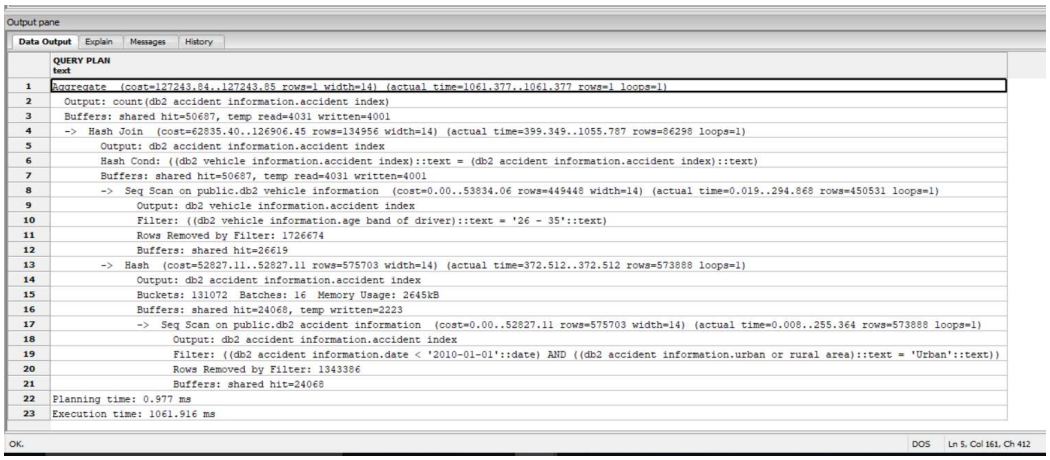
	QUERY PLAN text
1	Aggregate (cost=127243.84..127243.85 rows=1 width=14) (actual time=1067.389..1067.389 rows=1 loops=1)
2	Output: count(db2 accident information.accident index)
3	Buffers: shared hit=50687, temp read=4031 written=4001
4	-> Hash Join (cost=62835.40..126906.45 rows=134956 width=14) (actual time=410.424..1061.901 rows=86298 loops=1)
5	Output: db2 accident information.accident index
6	Hash Cond: ((db2 vehicle information.accident index)::text = (db2 accident information.accident index)::text)
7	Buffers: shared hit=50687, temp read=4031 written=4001
8	-> Seq Scan on public.db2 vehicle information (cost=0.00..53834.06 rows=449448 width=14) (actual time=0.009..287.774 rows=450531 loops=1)
9	Output: db2 vehicle information.accident index
10	Filter: ((db2 vehicle information.age band of driver)::text = '26 - 35'::text)
11	Rows Removed by Filter: 1726674
12	Buffers: shared hit=26619
13	-> Hash (cost=52827.11..52827.11 rows=575703 width=14) (actual time=383.183..383.183 rows=573888 loops=1)
14	Output: db2 accident information.accident index
15	Buckets: 131072 Batches: 16 Memory Usage: 2645kB
16	Buffers: shared hit=24068, temp written=2223
17	-> Seq Scan on public.db2 accident information (cost=0.00..52827.11 rows=575703 width=14) (actual time=0.007..267.355 rows=573888 loops=1)
18	Output: db2 accident information.accident index
19	Filter: ((db2 accident information.date < '2010-01-01'::date) AND ((db2 accident information.urban or rural area)::text = 'Urban'::text))
20	Rows Removed by Filter: 1343386
21	Buffers: shared hit=24068
22	Planning time: 0.941 ms
23	Execution time: 1068.562 ms

Planning time:0.941ms

Execution time:1068.562ms

2η Εκτέλεση(αυτή που μας ενδιαφέρει (αν και λέω δεύτερη εκτέλεση μπορεί να έχω τρέξει περισσότερες από τρείς φορές την επερώτηση))



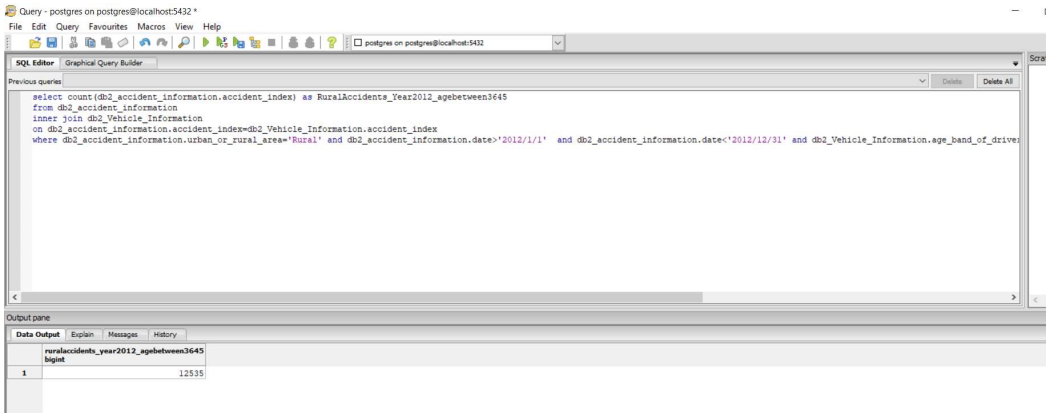


Planning time:0.977ms

Execution time:1061.916ms

time:1msecs

iv. Βρείτε πόσα ατυχήματα έγιναν σε αγροτική περιοχή το έτος 2012 και η ηλικιακή κατηγορία του οδηγού είναι 36-45.



select count(db2_accident_information.accident_index) as RuralAccidents_Year2012_agebetween3645

from db2_accident_information

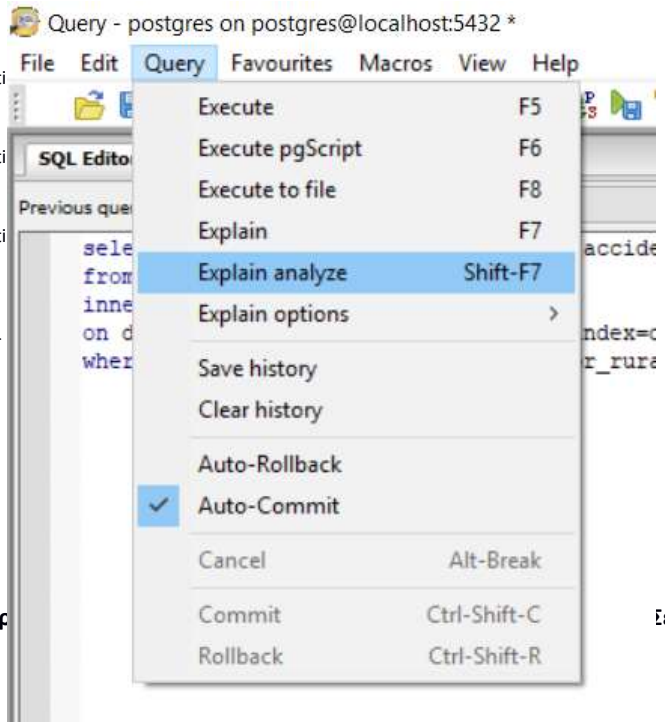
inner join db2_Vehicle_Information

on db2_accident_information.accident_index=db2_Vehicle_Information.accident_index

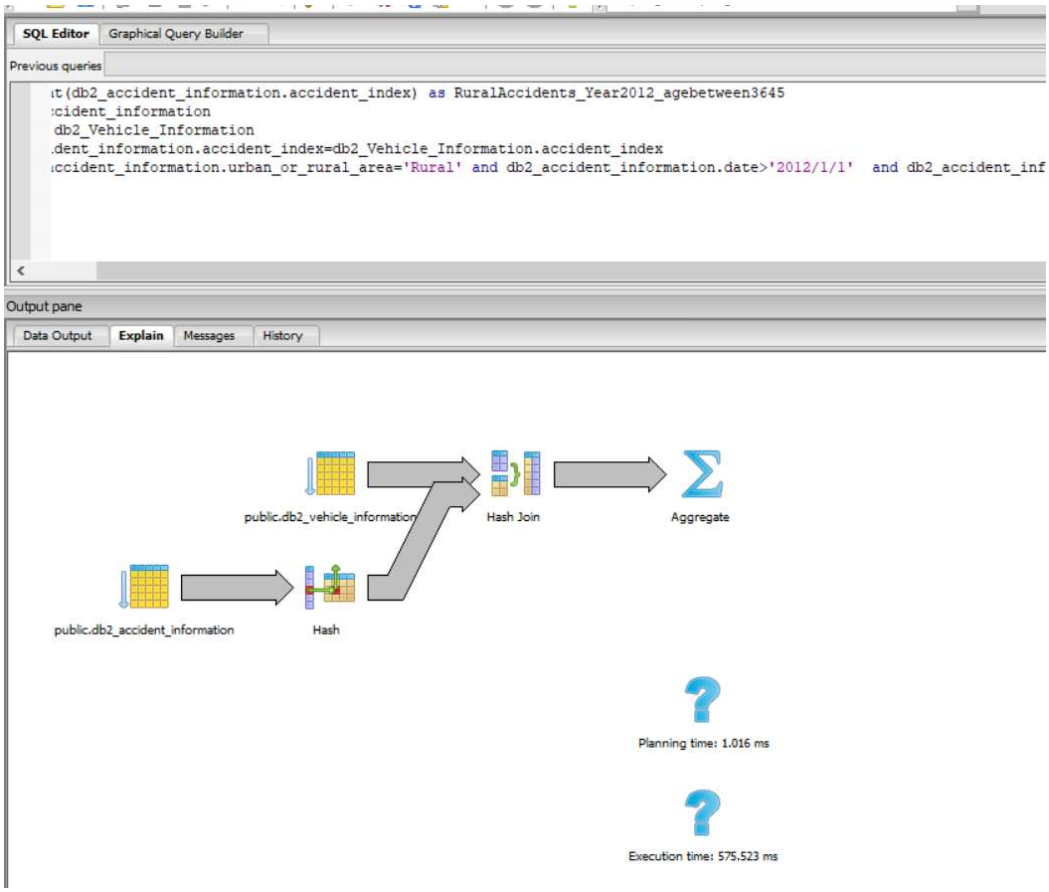
where db2_accident_information.urban_or_rural_area='Rural' and db2_accident_information.date>'2012/1/1' and db2_accident_information.date<'2012/12/31' and db2_Vehicle_Information.age_band_of_driver='36 – 45'

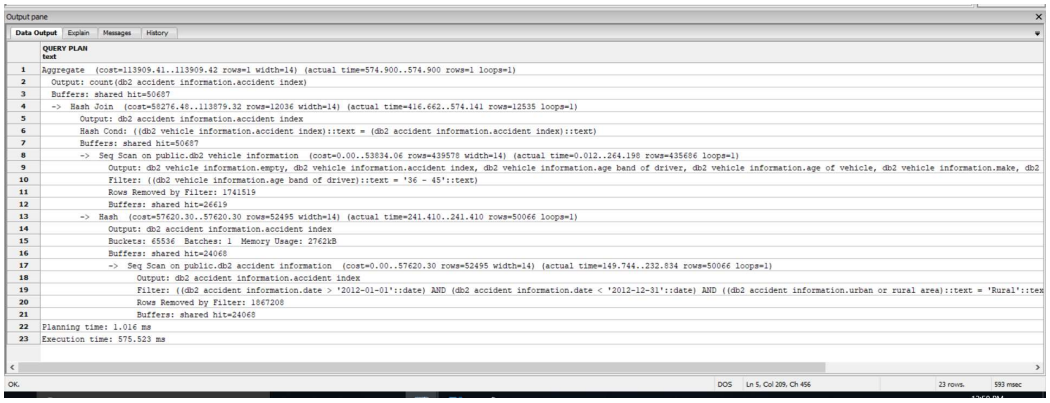
Κάνουμε select για να εμφανίσουμε πόσα ατυχήματα έγιναν σε αγροτική περιοχή το έτος 2012 και η ηλικιακή κατηγορία του οδηγού είναι 36-45.Κανουμε σύνδεση πινάκων (inner join)

where db2_accident_information.urban_or_rural_area='Rural' and db2_accident_information.date>'2012/1/1' and db2_accident_information.date<'2012/12/31' and db2_Vehicle_Information.age_band_of_driver='36 – 45'



1η Εκτέλεση


SQL Editor
Graphical Query Builder
Previous queries
ιτ(db2_accident_information.accident_index) as RuralAccidents_Year2012_agebetween3645
:cident_information
db2_Vehicle_Information
.dent_information.accident_index=db2_Vehicle_Information.accident_index
ccident_information.urban_or_rural_area='Rural' and db2_accident_information.date>'2012/1/1' and db2_accident_inf
Output pane
Data Output
Explain
Messages
History
public.db2_vehicle_information
Hash Join
Aggregate
public.db2_accident_information
Hash
Planning time: 1.016 ms
Execution time: 575.523 ms



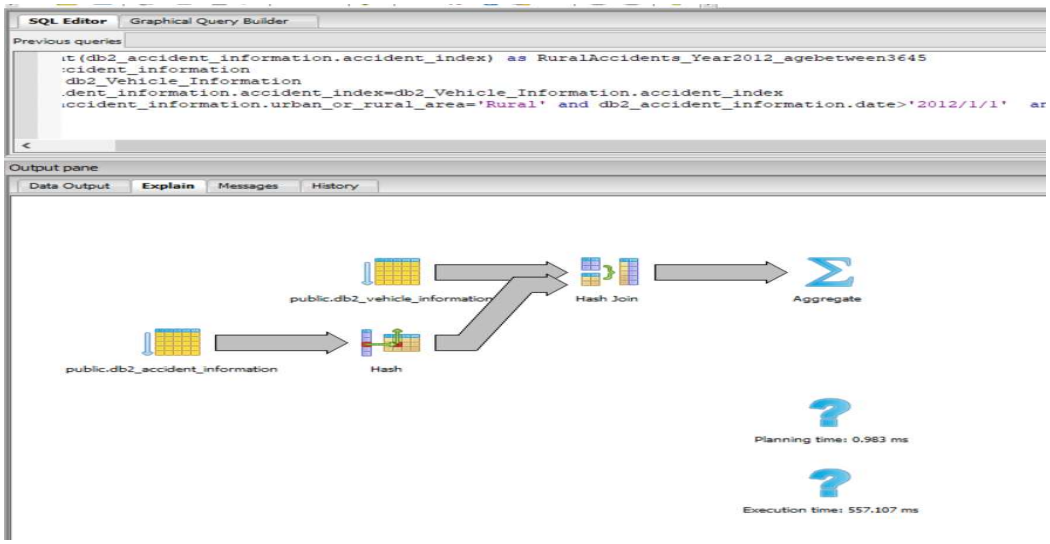
Output pane

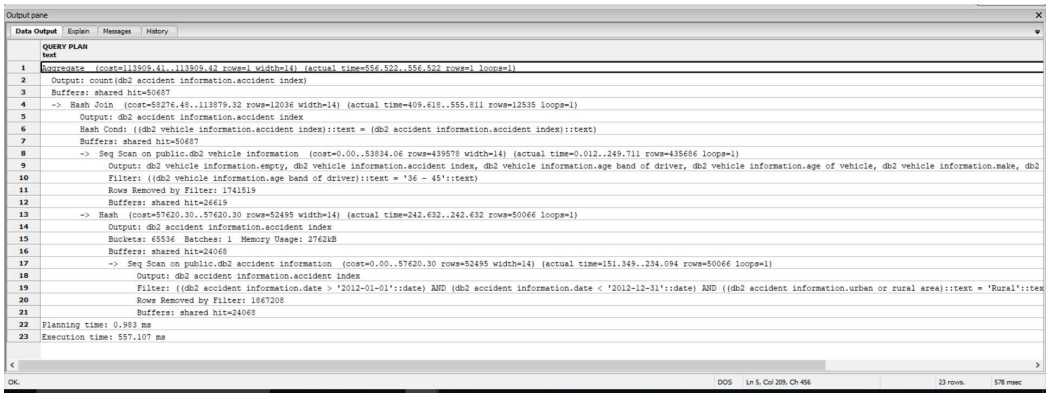
Data Output | Explain | Messages | History

	QUERY PLAN text
1	Aggregate (cost=113909.41..113909.42 rows=1 width=14) (actual time=574.900..574.900 rows=1 loops=1)
2	Output: count(db2 accident information.accident index)
3	Buffers: shared hit=50687
4	-> Hash Join (cost=58276.48..113879.32 rows=12036 width=14) (actual time=416.662..574.141 rows=12535 loops=1)
5	Output: db2 accident information.accident index
6	Hash Cond: ((db2 vehicle information.accident index)::text = (db2 accident information.accident index)::text)
7	Buffers: shared hit=50687
8	-> Seq Scan on public.db2 vehicle information (cost=0.00..53834.06 rows=439578 width=14) (actual time=0.012..264.198 rows=435686 loops=1)
9	Output: db2 vehicle information.empty, db2 vehicle information.accident index, db2 vehicle information.age band of driver, db2 vehicle information.age of vehicle, db2 vehicle information.make, db2
10	Filter: ((db2 vehicle information.age band of driver)::text = '36 - 45'::text)
11	Rows Removed by Filter: 1741519
12	Buffers: shared hit=26619
13	-> Hash (cost=57620.30..57620.30 rows=52495 width=14) (actual time=241.410..241.410 rows=50066 loops=1)
14	Output: db2 accident information.accident index
15	Buckets: 65536 Batches: 1 Memory Usage: 2762kB
16	Buffers: shared hit=24068
17	-> Seq Scan on public.db2 accident information (cost=0.00..57620.30 rows=52495 width=14) (actual time=149.744..232.834 rows=50066 loops=1)
18	Output: db2 accident information.accident index
19	Filter: ((db2 accident information.date > '2012-01-01'::date) AND (db2 accident information.date < '2012-12-31'::date) AND ((db2 accident information.urban or rural area)::text = 'Rural'::tex
20	Rows Removed by Filter: 1867208
21	Buffers: shared hit=24068
22	Planning time: 1.016 ms
23	Execution time: 575.523 ms

OK. DOS Ln 5, Col 209, Ch 456 23 rows. 593 msec

2η Εκτέλεση(αυτή που μας ενδιαφέρει και κρατάμε(αν και λέω δεύτερη εκτέλεση μπορεί να έχω τρέξει περισσότερες από τρείς φορές την επερώτηση))





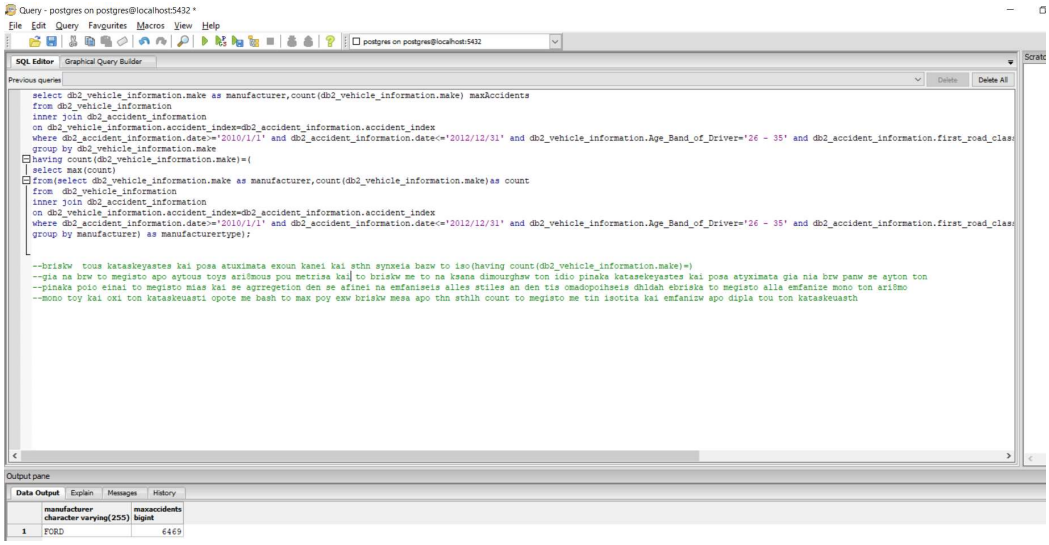
```
QUERY PLAN
text
Aggregate  (cost=113909.41..113909.42 rows=1 width=14) (actual time=556.522..556.522 rows=1 loops=1)
  Output: count(db2_accident_information.accident_index)
  Buffers: shared hit=50687
  ->  Hash Join  (cost=58276.48..113879.32 rows=12036 width=14) (actual time=409.618..555.811 rows=12535 loops=1)
        Output: db2_accident_information.accident_index
        Hash Cond: ((db2_vehicle_information.accident_index)::text = (db2_accident_information.accident_index)::text)
        Buffers: shared hit=50687
        ->  Seq Scan on public.db2_vehicle_information  (cost=0.00..53834.06 rows=439578 width=14) (actual time=0.012..249.711 rows=435686 loops=1)
              Output: db2_vehicle_information.empty, db2_vehicle_information.accident_index, db2_vehicle_information.age_band_of_driver, db2_vehicle_information.age_of_vehicle, db2_vehicle_information.make, db2
              Filter: ((db2_vehicle_information.age_band_of_driver)::text = '36 - 45'::text)
              Rows Removed by Filter: 1741519
              Buffers: shared hit=26619
        ->  Hash  (cost=57620.30..57620.30 rows=52495 width=14) (actual time=242.632..242.632 rows=50066 loops=1)
              Output: db2_accident_information.accident_index
              Buckets: 65536  Batches: 1  Memory Usage: 2762kB
              Buffers: shared hit=24068
              ->  Seq Scan on public.db2_accident_information  (cost=0.00..57620.30 rows=52495 width=14) (actual time=151.349..234.094 rows=50066 loops=1)
                    Output: db2_accident_information.accident_index
                    Filter: ((db2_accident_information.date > '2012-01-01'::date) AND (db2_accident_information.date < '2012-12-31'::date) AND ((db2_accident_information.urban_or_rural_area)::text = 'Rural'::tex
                    Rows Removed by Filter: 1867208
                    Buffers: shared hit=24068
Planning time: 0.983 ms
Execution time: 557.107 ms
```

Planning time:0.983ms

Execution time:557.107ms

time:578msecs

v. Βρείτε τον κατασκευαστή του οχήματος που έχει τα περισσότερα ατυχήματα μεταξύ των ετών 2010 και 2012, η ηλικιακή κατηγορία των οδηγών είναι 26- 35 και η κατηγορία δρόμου είναι Α.



select db2_vehicle_information.make as manufacturer,count(db2_vehicle_information.make) maxAccidents

from db2_vehicle_information

inner join db2_accident_information

on db2_vehicle_information.accident_index=db2_accident_information.accident_index

where db2_accident_information.date>='2010/1/1' and db2_accident_information.date<='2012/12/31' and db2_vehicle_information.Age_Band_of_Driver='26 - 35' and db2_accident_information.first_road_class='A'

group by db2_vehicle_information.make

having count(db2_vehicle_information.make)=(

select max(count)

from(select db2_vehicle_information.make as manufacturer,count(db2_vehicle_information.make)as count

from db2_vehicle_information

inner join db2_accident_information

on db2_vehicle_information.accident_index=db2_accident_information.accident_index

where db2_accident_information.date>='2010/1/1' and db2_accident_information.date<='2012/12/31' and db2_vehicle_information.Age_Band_of_Driver='26 - 35' and db2_accident_information.first_road_class='A'

group by manufacturer) as manufacturertype);

Βρίσκω τους κατασκευαστές και πόσα ατυχήματα έχουν κάνει. Επιλέγοντας από τους δυο πίνακες κατασκευαστή (**make**) και μετρώντας πόσες φορές εμφανίζεται ο κατασκευαστής με **count(make)** κάνοντας σύνδεση στους δυο πίνακες γιατί θέλω να είναι τα ατυχήματα μεταξύ των ετών 2010 και 2012, η ηλικιακή κατηγορία των οδηγών είναι 26- 35 και η κατηγορία δρόμου είναι Α ομαδοποιώντας με βάση τον κατασκευαστή **group by make** άρα

select db2_vehicle_information.make as manufacturer,count(db2_vehicle_information.make) maxAccidents

from db2_vehicle_information

inner join db2_accident_information

on db2_vehicle_information.accident_index=db2_accident_information.accident_index

where db2_accident_information.date>='2010/1/1' and db2_accident_information.date<='2012/12/31' and db2_vehicle_information.Age_Band_of_Driver='26 - 35' and db2_accident_information.first_road_class='A'

group by db2_vehicle_information.make

και στην συνέχεια βάζω το ίσο(having count(db2_vehicle_information.make)=) από την στήλη με τους αριθμούς που είδη έχω βρει και το συγκρίνω με το μέγιστο που βρισκω στην συνέχεια δηλαδή

having count(db2_vehicle_information.make)=(select max(count)

για να βρω το μέγιστο από αυτούς τους αριθμούς και το βρίσκω με το να ξανά δημιουργήσω όπως έκανα και στην αρχή τον ίδιο πίνακα κατασκευαστές και πόσα ατυχήματα έκανε ο κάθε κατασκευαστής με τους ίδιους περιορισμού και την ίδια ομαδοποίηση για να βρω πάνω σε αυτόν τον πίνακα ποιος είναι ο μέγιστος αριθμός μιας και σε aggregation δεν σε αφήνει να εμφανίσεις άλλες στήλες αν δεν τις ομαδοποιήσεις . Δηλαδή έβρισκα τον μέγιστο αλλά εμφάνιζε μόνο τον αριθμό μόνο του και όχι και τον κατασκευαστή οπότε με βάση το μέγιστο (max) που βρίσκω μέσα από την στήλη count βρισκω το μέγιστο και μετά με βάση την ισότητα που έχω βάλει παραπάνω having count(db2_vehicle_information.make)=(select max(count) βρισκω από την αρχική στήλη count το μέγιστο και εμφανίζω από δίπλα του τον κατασκευαστή.

having count(db2_vehicle_information.make)=(select max(count)

from(select db2_vehicle_information.make as manufacturer,count(db2_vehicle_information.make)as count

from db2_vehicle_information

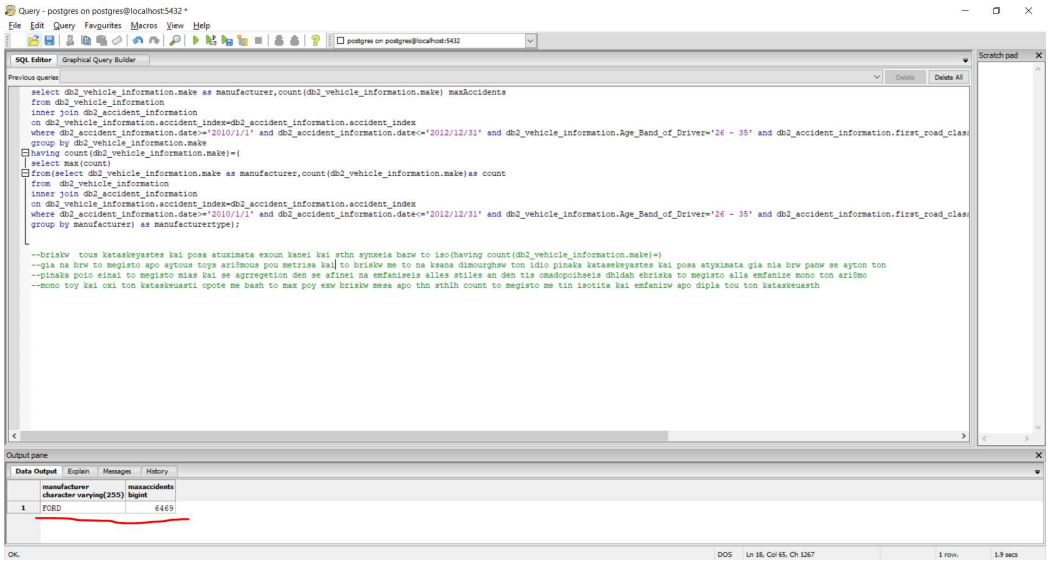
inner join db2_accident_information

on db2_vehicle_information.accident_index=db2_accident_information.accident_index

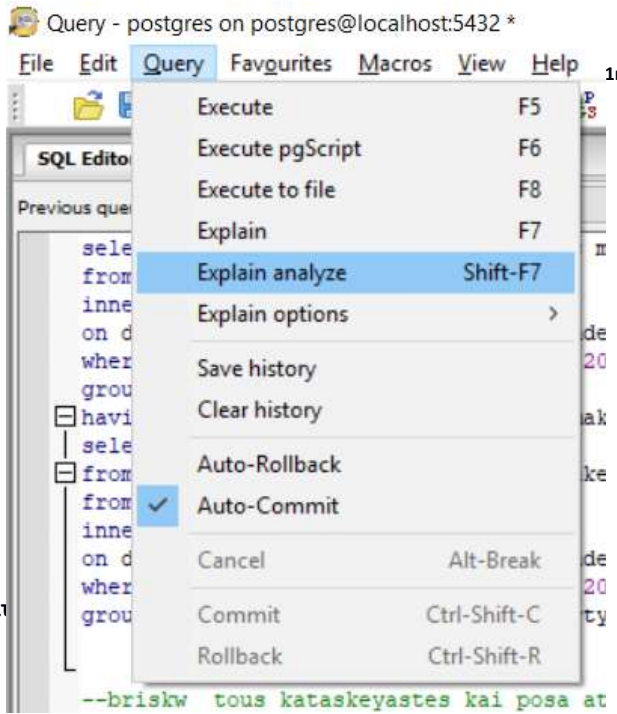
where db2_accident_information.date>='2010/1/1' and db2_accident_information.date<='2012/12/31' and db2_vehicle_information.Age_Band_of_Driver='26 - 35' and db2_accident_information.first_road_class='A'

group by manufacturer) as manufacturertype);

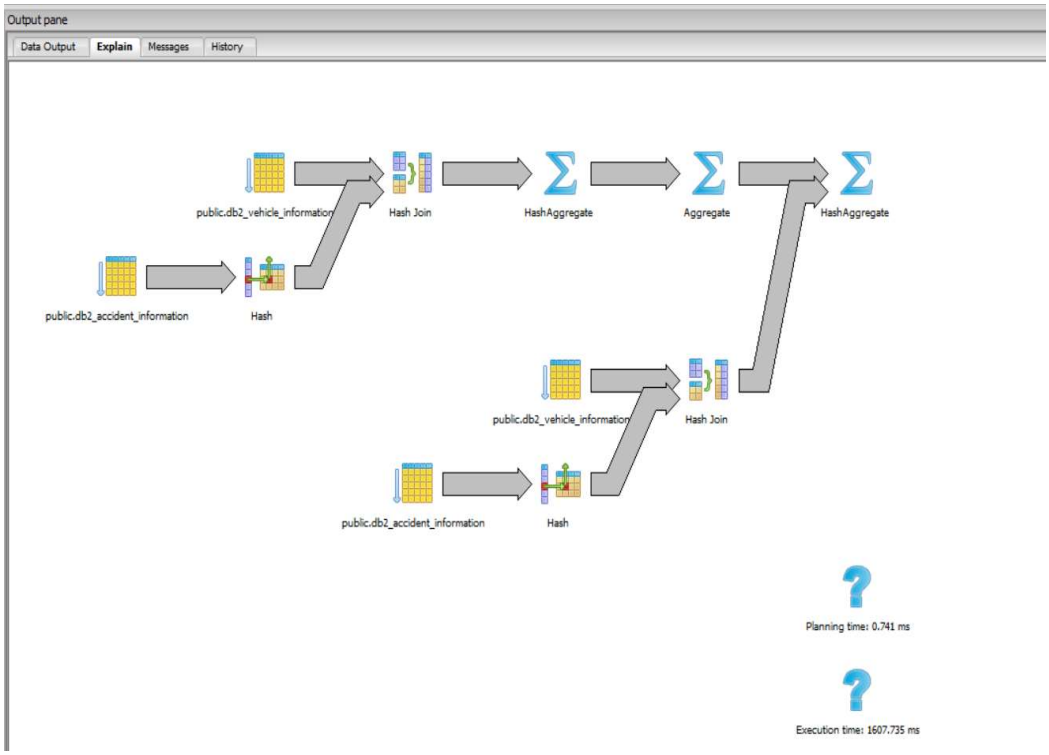
Βλέπουμε τον μέγιστο που εμφανίζει



Ο καλύτερος κατασκευαστής είναι ο **FORD 6469 ατυχήματα**

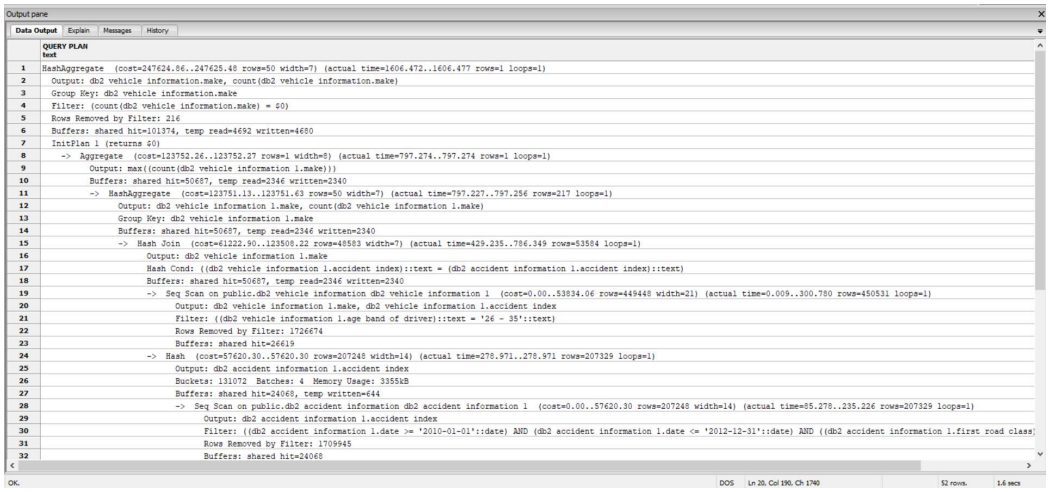


1η εκτέλεση

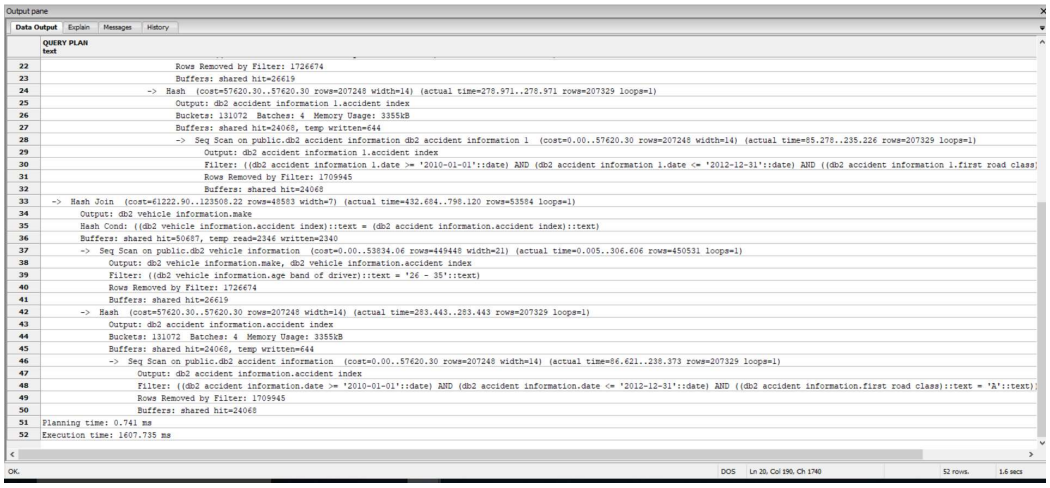


Planning time: 0.741 ms

Execution time: 1607.735 ms



```
Output pane
Data Output | Explain | Messages | History

QUERY PLAN
text
1  HashAggregate  (cost=247624.86..247625.48 rows=50 width=7) (actual time=1606.472..1606.477 rows=1 loops=1)
2    Output: db2 vehicle information.make, count(db2 vehicle information.make)
3    Group Key: db2 vehicle information.make
4    Filter: (count(db2 vehicle information.make) = 40)
5    Rows Removed by Filter: 216
6    Buffers: shared hit=101374, temp read=4692 written=4680
7    InitPlan 1 (returns $0)
8      ->  Aggregate  (cost=123752.26..123752.27 rows=1 width=8) (actual time=797.274..797.274 rows=1 loops=1)
9            Output: max((count(db2 vehicle information 1.make)))
10           Buffers: shared hit=50687, temp read=2346 written=2340
11           ->  HashAggregate  (cost=123751.13..123751.63 rows=50 width=7) (actual time=797.227..797.256 rows=217 loops=1)
12                 Output: db2 vehicle information 1.make, count(db2 vehicle information 1.make)
13                 Group Key: db2 vehicle information 1.make
14                 Buffers: shared hit=50687, temp read=2346 written=2340
15                 ->  Hash Join  (cost=61222.90..123508.22 rows=48583 width=7) (actual time=429.235..786.349 rows=53584 loops=1)
16                       Output: db2 vehicle information 1.make
17                       Hash Cond: ((db2 vehicle information 1.accident index)::text = (db2 accident information 1.accident index)::text)
18                       Buffers: shared hit=50687, temp read=2346 written=2340
19                       ->  Seq Scan on public.db2 vehicle information db2 vehicle information 1  (cost=0.00..53834.06 rows=449448 width=21) (actual time=0.009..300.780 rows=450531 loops=1)
20                             Output: db2 vehicle information 1.make, db2 vehicle information 1.accident index
21                             Filter: ((db2 vehicle information 1.age band of driver)::text = '26 - 35'::text)
22                             Rows Removed by Filter: 1726674
23                             Buffers: shared hit=26619
24                       ->  Hash  (cost=57620.30..57620.30 rows=207248 width=14) (actual time=278.971..278.971 rows=207329 loops=1)
25                             Output: db2 accident information 1.accident index
26                             Buckets: 131072  Batches: 4  Memory Usage: 3355kB
27                             Buffers: shared hit=24068, temp written=644
28                             ->  Seq Scan on public.db2 accident information db2 accident information 1  (cost=0.00..57620.30 rows=207248 width=14) (actual time=85.278..235.226 rows=207329 loops=1)
29                                   Output: db2 accident information 1.accident index
30                                   Filter: ((db2 accident information 1.date >= '2010-01-01'::date) AND (db2 accident information 1.date <= '2012-12-31'::date) AND ((db2 accident information 1.first road class
31                                   Rows Removed by Filter: 1709945
32                                   Buffers: shared hit=24068
OK.                                                                                  DOS  Ln 20, Col 190, Ch 1740                    52 rows.  1.6 secs
```

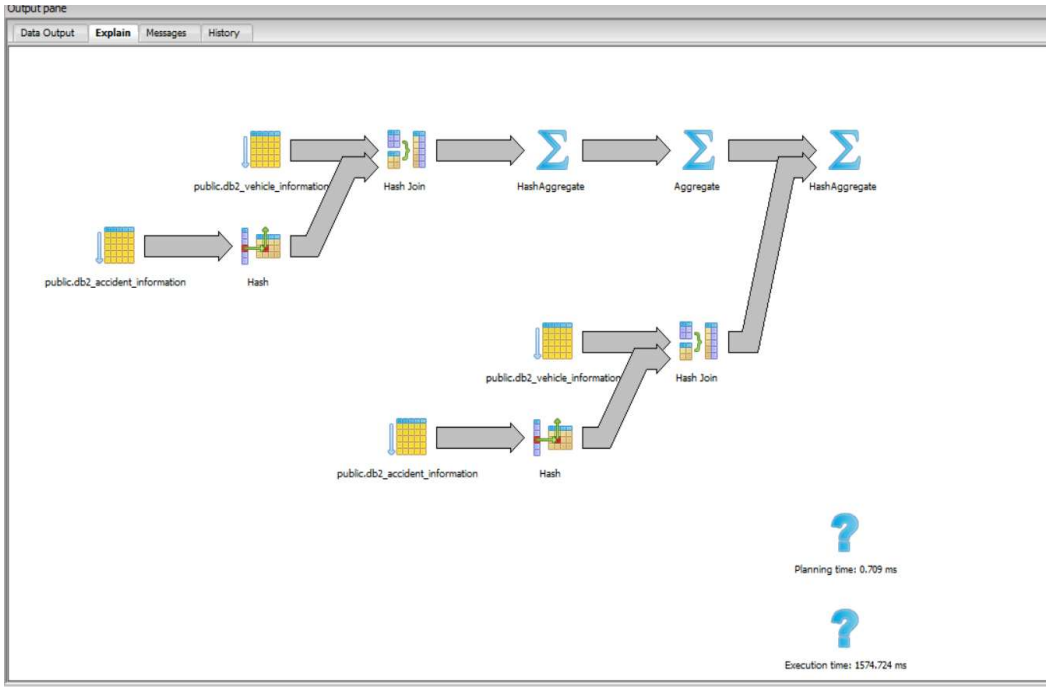


```
Output pane
Data Output | Explain | Messages | History

QUERY PLAN
text
22                                   Rows Removed by Filter: 1726674
23                                   Buffers: shared hit=26619
24                             ->  Hash  (cost=57620.30..57620.30 rows=207248 width=14) (actual time=278.971..278.971 rows=207329 loops=1)
25                                   Output: db2 accident information 1.accident index
26                                   Buckets: 131072  Batches: 4  Memory Usage: 3355kB
27                                   Buffers: shared hit=24068, temp written=644
28                                   ->  Seq Scan on public.db2 accident information db2 accident information 1  (cost=0.00..57620.30 rows=207248 width=14) (actual time=85.278..235.226 rows=207329 loops=1)
29                                         Output: db2 accident information 1.accident index
30                                         Filter: ((db2 accident information 1.date >= '2010-01-01'::date) AND (db2 accident information 1.date <= '2012-12-31'::date) AND ((db2 accident information 1.first road class
31                                         Rows Removed by Filter: 1709945
32                                         Buffers: shared hit=24068
33     ->  Hash Join  (cost=61222.90..123508.22 rows=48583 width=7) (actual time=432.684..798.120 rows=53584 loops=1)
34           Output: db2 vehicle information.make
35           Hash Cond: ((db2 vehicle information.accident index)::text = (db2 accident information.accident index)::text)
36           Buffers: shared hit=50687, temp read=2346 written=2340
37           ->  Seq Scan on public.db2 vehicle information  (cost=0.00..53834.06 rows=449448 width=21) (actual time=0.005..306.606 rows=450531 loops=1)
38                 Output: db2 vehicle information.make, db2 vehicle information.accident index
39                 Filter: ((db2 vehicle information.age band of driver)::text = '26 - 35'::text)
40                 Rows Removed by Filter: 1726674
41                 Buffers: shared hit=26619
42           ->  Hash  (cost=57620.30..57620.30 rows=207248 width=14) (actual time=283.443..283.443 rows=207329 loops=1)
43                 Output: db2 accident information.accident index
44                 Buckets: 131072  Batches: 4  Memory Usage: 3355kB
45                 Buffers: shared hit=24068, temp written=644
46                 ->  Seq Scan on public.db2 accident information  (cost=0.00..57620.30 rows=207248 width=14) (actual time=86.621..238.373 rows=207329 loops=1)
47                       Output: db2 accident information.accident index
48                       Filter: ((db2 accident information.date >= '2010-01-01'::date) AND (db2 accident information.date <= '2012-12-31'::date) AND ((db2 accident information.first road class)::text = 'A'::text)
49                       Rows Removed by Filter: 1709945
50                       Buffers: shared hit=24068
51 Planning time: 0.741 ms
52 Execution time: 1607.735 ms
OK.                                                                                  DOS  Ln 20, Col 190, Ch 1740                    52 rows.  1.6 secs
```

Planning time:0.741 ms

Execution time:1607.735ms

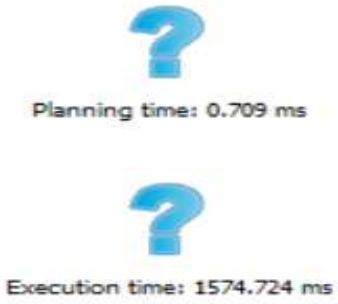
2η Εκτέλεση(αυτή που μας ενδιαφέρει και κρατάμε(αν και λέω δεύτερη εκτέλεση μπορεί να έχω τρέξει περισσότερες από τρείς φορές την επερώτηση))

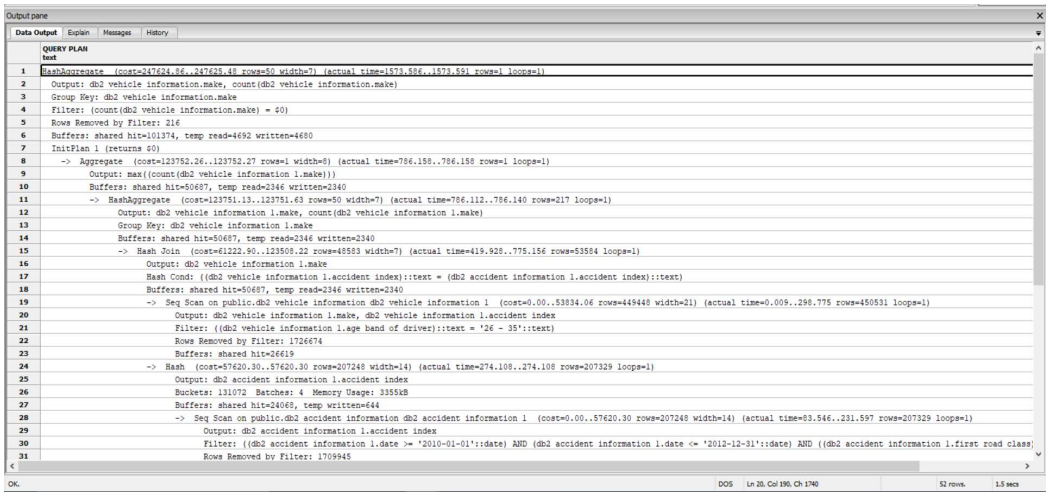


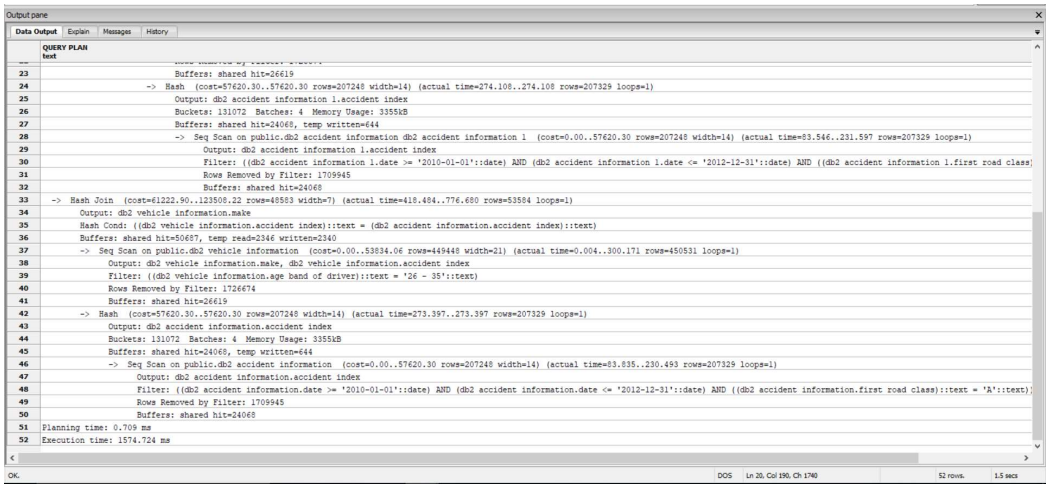
Planning time 0.709ms

Execution time:1574.724ms

time:1.5secs




Output pane
Data Output
Explain
Messages
History
QUERY PLAN
text
HashAggregate (cost=247624.86..247625.48 rows=50 width=7) (actual time=1573.586..1573.591 rows=1 loops=1)
Output: db2 vehicle information.make, count(db2 vehicle information.make)
Group Key: db2 vehicle information.make
Filter: (count(db2 vehicle information.make) = $0)
Rows Removed by Filter: 216
Buffers: shared hit=101374, temp read=4692 written=4680
InitPlan 1 (returns $0)
-> Aggregate (cost=123752.26..123752.27 rows=1 width=8) (actual time=786.158..786.158 rows=1 loops=1)
Output: max((count(db2 vehicle information 1.make)))
Buffers: shared hit=50687, temp read=2346 written=2340
-> HashAggregate (cost=123751.13..123751.63 rows=50 width=7) (actual time=786.112..786.140 rows=217 loops=1)
Output: db2 vehicle information 1.make, count(db2 vehicle information 1.make)
Group Key: db2 vehicle information 1.make
Buffers: shared hit=50687, temp read=2346 written=2340
-> Hash Join (cost=61222.90..123508.22 rows=48583 width=7) (actual time=419.928..775.156 rows=53584 loops=1)
Output: db2 vehicle information 1.make
Hash Cond: ((db2 vehicle information 1.accident index)::text = (db2 accident information 1.accident index)::text)
Buffers: shared hit=50687, temp read=2346 written=2340
-> Seq Scan on public.db2 vehicle information db2 vehicle information 1 (cost=0.00..53834.06 rows=449448 width=21) (actual time=0.009..298.775 rows=450531 loops=1)
Output: db2 vehicle information 1.make, db2 vehicle information 1.accident index
Filter: ((db2 vehicle information 1.age band of driver)::text = '26 - 35'::text)
Rows Removed by Filter: 1726674
Buffers: shared hit=26619
-> Hash (cost=57620.30..57620.30 rows=207248 width=14) (actual time=274.108..274.108 rows=207329 loops=1)
Output: db2 accident information 1.accident index
Buckets: 131072 Batches: 4 Memory Usage: 3355kB
Buffers: shared hit=24068, temp written=644
-> Seq Scan on public.db2 accident information db2 accident information 1 (cost=0.00..57620.30 rows=207248 width=14) (actual time=83.546..231.597 rows=207329 loops=1)
Output: db2 accident information 1.accident index
Filter: ((db2 accident information 1.date >= '2010-01-01'::date) AND (db2 accident information 1.date <= '2012-12-31'::date) AND ((db2 accident information 1.first road class
Rows Removed by Filter: 1709945
OK.
DOS
Ln 20, Col 190, Ch 1740
52 rows.
1.5 secs


Output pane
Data Output
Explain
Messages
History
QUERY PLAN
text
Buffers: shared hit=26619
-> Hash (cost=57620.30..57620.30 rows=207248 width=14) (actual time=274.108..274.108 rows=207329 loops=1)
Output: db2 accident information 1.accident index
Buckets: 131072 Batches: 4 Memory Usage: 3355kB
Buffers: shared hit=24068, temp written=644
-> Seq Scan on public.db2 accident information db2 accident information 1 (cost=0.00..57620.30 rows=207248 width=14) (actual time=83.546..231.597 rows=207329 loops=1)
Output: db2 accident information 1.accident index
Filter: ((db2 accident information 1.date >= '2010-01-01'::date) AND (db2 accident information 1.date <= '2012-12-31'::date) AND ((db2 accident information 1.first road class
Rows Removed by Filter: 1709945
Buffers: shared hit=24068
-> Hash Join (cost=61222.90..123508.22 rows=48583 width=7) (actual time=418.484..776.680 rows=53584 loops=1)
Output: db2 vehicle information.make
Hash Cond: ((db2 vehicle information.accident index)::text = (db2 accident information.accident index)::text)
Buffers: shared hit=50687, temp read=2346 written=2340
-> Seq Scan on public.db2 vehicle information (cost=0.00..53834.06 rows=449448 width=21) (actual time=0.004..300.171 rows=450531 loops=1)
Output: db2 vehicle information.make, db2 vehicle information.accident index
Filter: ((db2 vehicle information.age band of driver)::text = '26 - 35'::text)
Rows Removed by Filter: 1726674
Buffers: shared hit=26619
-> Hash (cost=57620.30..57620.30 rows=207248 width=14) (actual time=273.397..273.397 rows=207329 loops=1)
Output: db2 accident information.accident index
Buckets: 131072 Batches: 4 Memory Usage: 3355kB
Buffers: shared hit=24068, temp written=644
-> Seq Scan on public.db2 accident information (cost=0.00..57620.30 rows=207248 width=14) (actual time=83.835..230.493 rows=207329 loops=1)
Output: db2 accident information.accident index
Filter: ((db2 accident information.date >= '2010-01-01'::date) AND (db2 accident information.date <= '2012-12-31'::date) AND ((db2 accident information.first road class)::text = 'A'::text)
Rows Removed by Filter: 1709945
Buffers: shared hit=24068
Planning time: 0.709 ms
Execution time: 1574.724 ms
OK.
DOS
Ln 20, Col 190, Ch 1740
52 rows.
1.5 secs

planning time:0.709ms

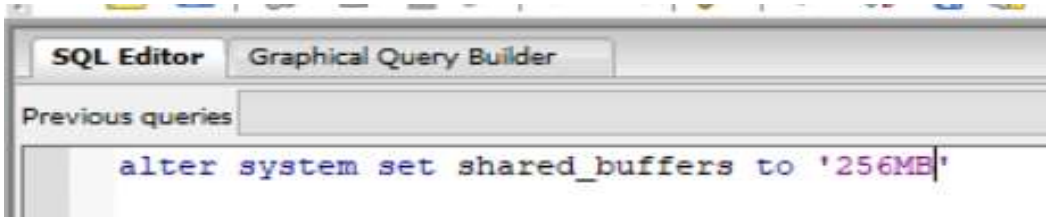
Execution time: 1574.724ms

time:1.5secs

(b) Ρυθμίστε την PostgreSQL έτσι ώστε να χρησιμοποιεί ως buffer περισσότερη μνήμη από τη μνήμη RAM του υπολογιστή σας (ικανή ώστε να χωράει όσο γίνεται περισσότερο από το dataset, όλο αν είναι δυνατόν). Έπειτα, εκτελέστε πάλι τις παραπάνω επερωτήσεις και εξηγήστε τι παρατηρείτε. TIP: shared_buffers (π.χ. ALTER SYSTEM SET shared_buffers TO '256MB'; -- απαιτείται επανεκκίνηση του postgreSQL server).

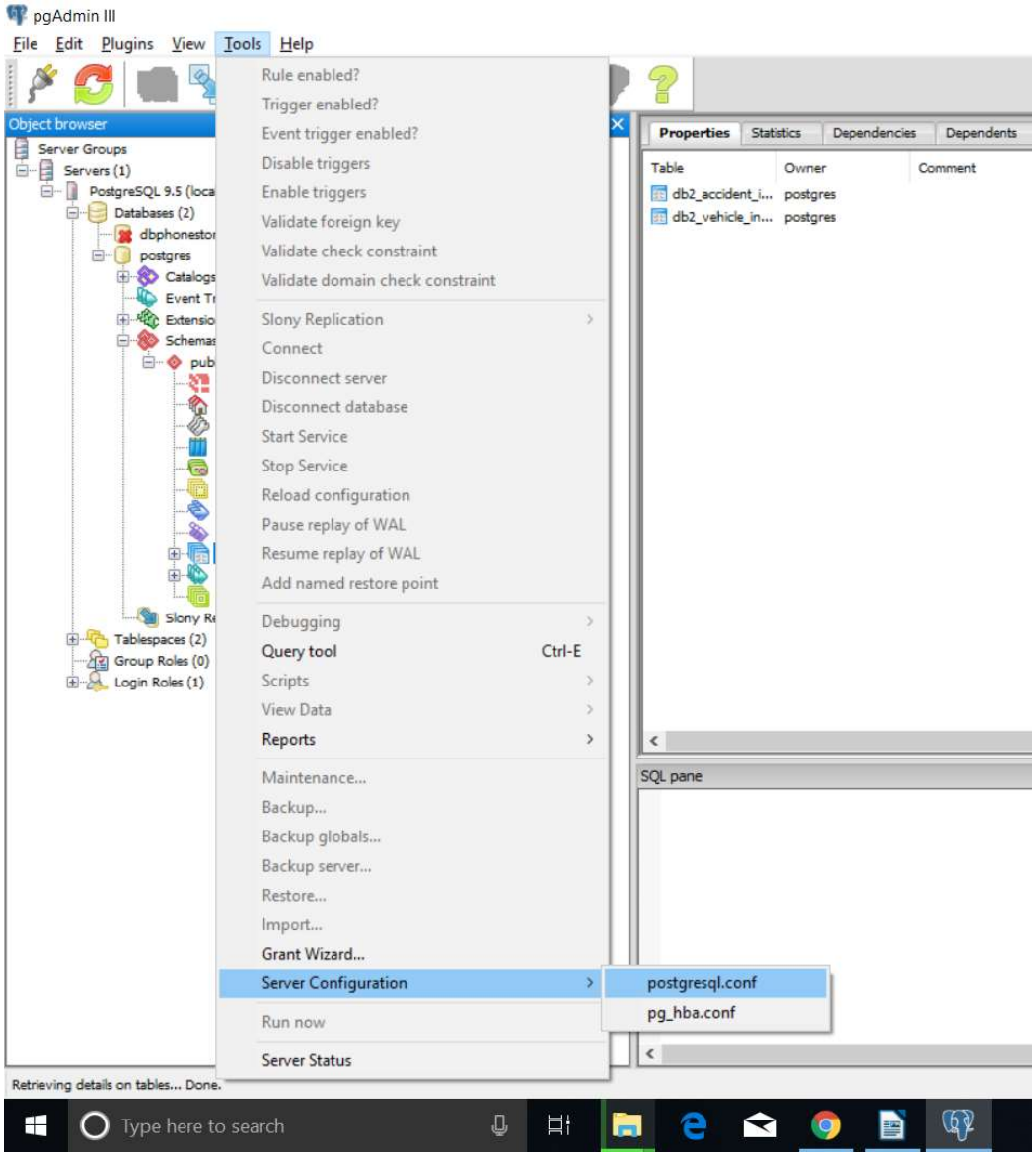
Λύση:

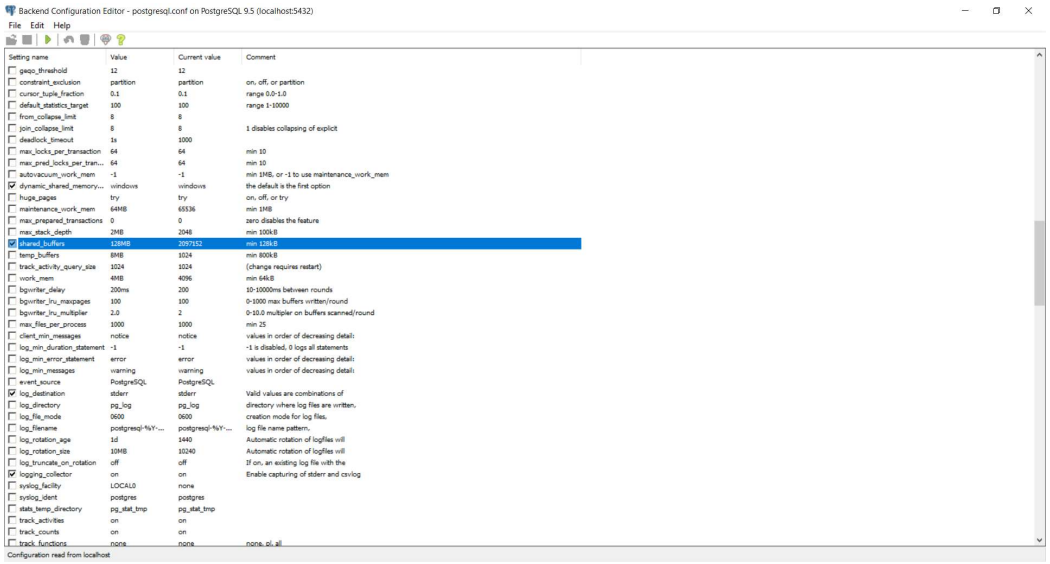
Αυτήν την ρύθμιση γίνεται με δυο τρόπους ο πρώτος τρόπος είναι να ανοίξουμε ένα query και να πληκτρολογήσουμε την εντολή.



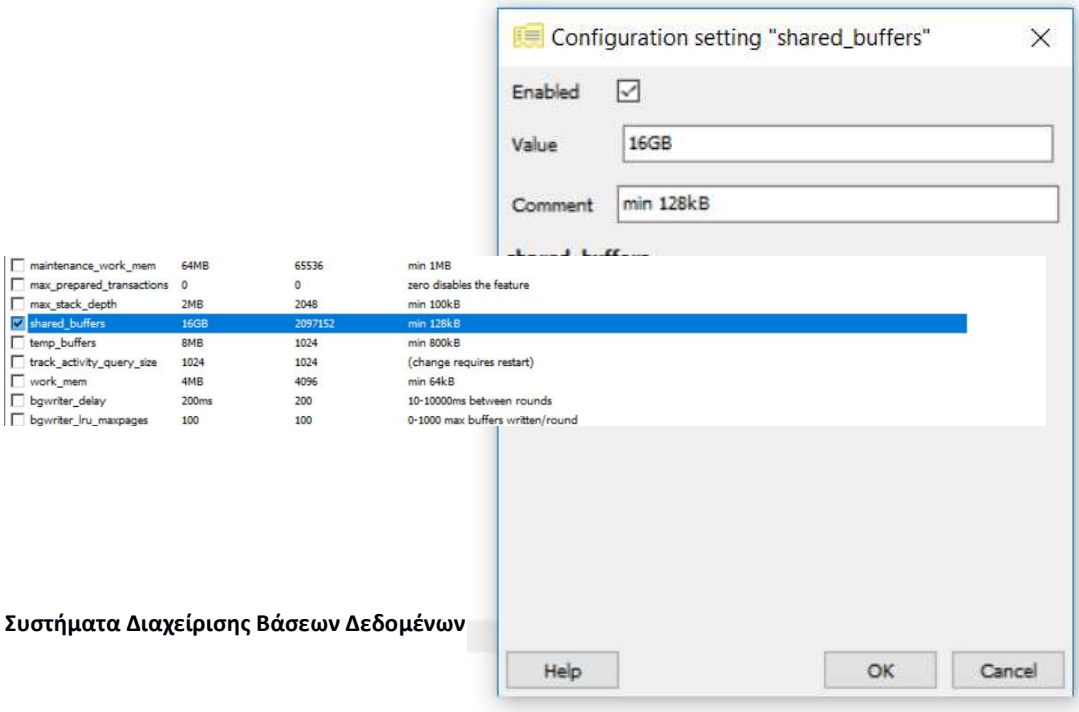
Alter system set shared_buffers to ‘256MB’

Ο δεύτερος τρόπος είναι να επιλέξουμε εμείς tools→ Server Configuration→ postgresql.conf


pgAdmin III
File Edit Plugins View Tools Help
Object browser
Server Groups
Servers (1)
PostgreSQL 9.5 (loca
Databases (2)
dbphonestor
postgres
Catalogs
Event Tr
Extensio
Schemas
pub
Slony Re
Tablespaces (2)
Group Roles (0)
Login Roles (1)
Rule enabled?
Trigger enabled?
Event trigger enabled?
Disable triggers
Enable triggers
Validate foreign key
Validate check constraint
Validate domain check constraint
Slony Replication
Connect
Disconnect server
Disconnect database
Start Service
Stop Service
Reload configuration
Pause replay of WAL
Resume replay of WAL
Add named restore point
Debugging
Query tool Ctrl-E
Scripts
View Data
Reports
Maintenance...
Backup...
Backup globals...
Backup server...
Restore...
Import...
Grant Wizard...
Server Configuration
Run now
Server Status
postgresql.conf
pg_hba.conf
Properties Statistics Dependencies Dependents
Table Owner Comment
db2_accident_i... postgres
db2_vehicle_in... postgres
SQL pane
Retrieving details on tables... Done.
Type here to search

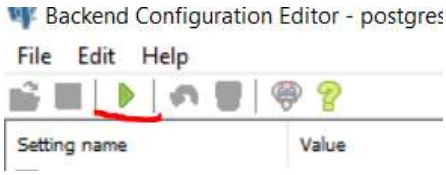


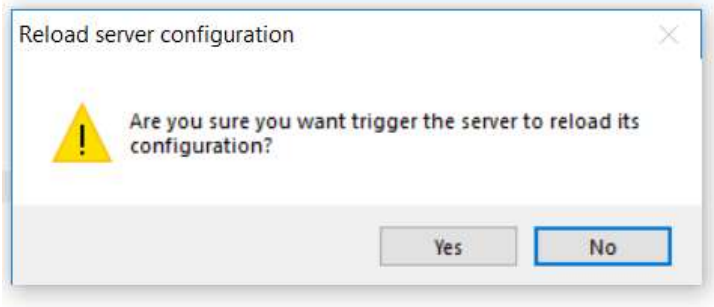
Θέτω το shared_buffers 16GB όσο είναι και η μνήμη ram μου



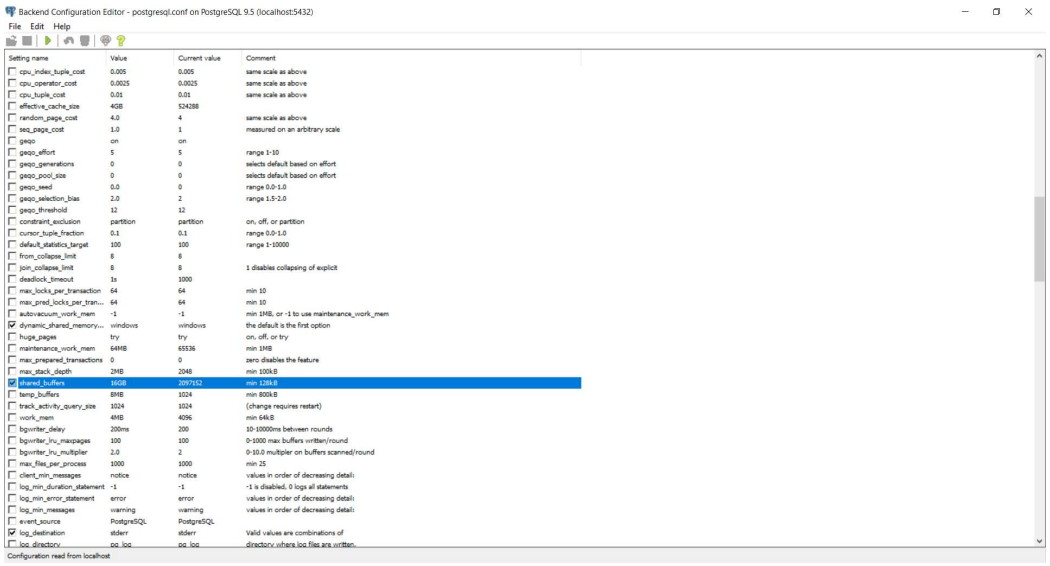
Και το ρυθμίζουμε στα 16 GB λίγο παραπάνω από την μνήμη ram.

Στην συνέχεια πατάμε να επιβεβαιώσουμε τις αλλαγές.





Και κάνουμε restart στον Postgres sql server



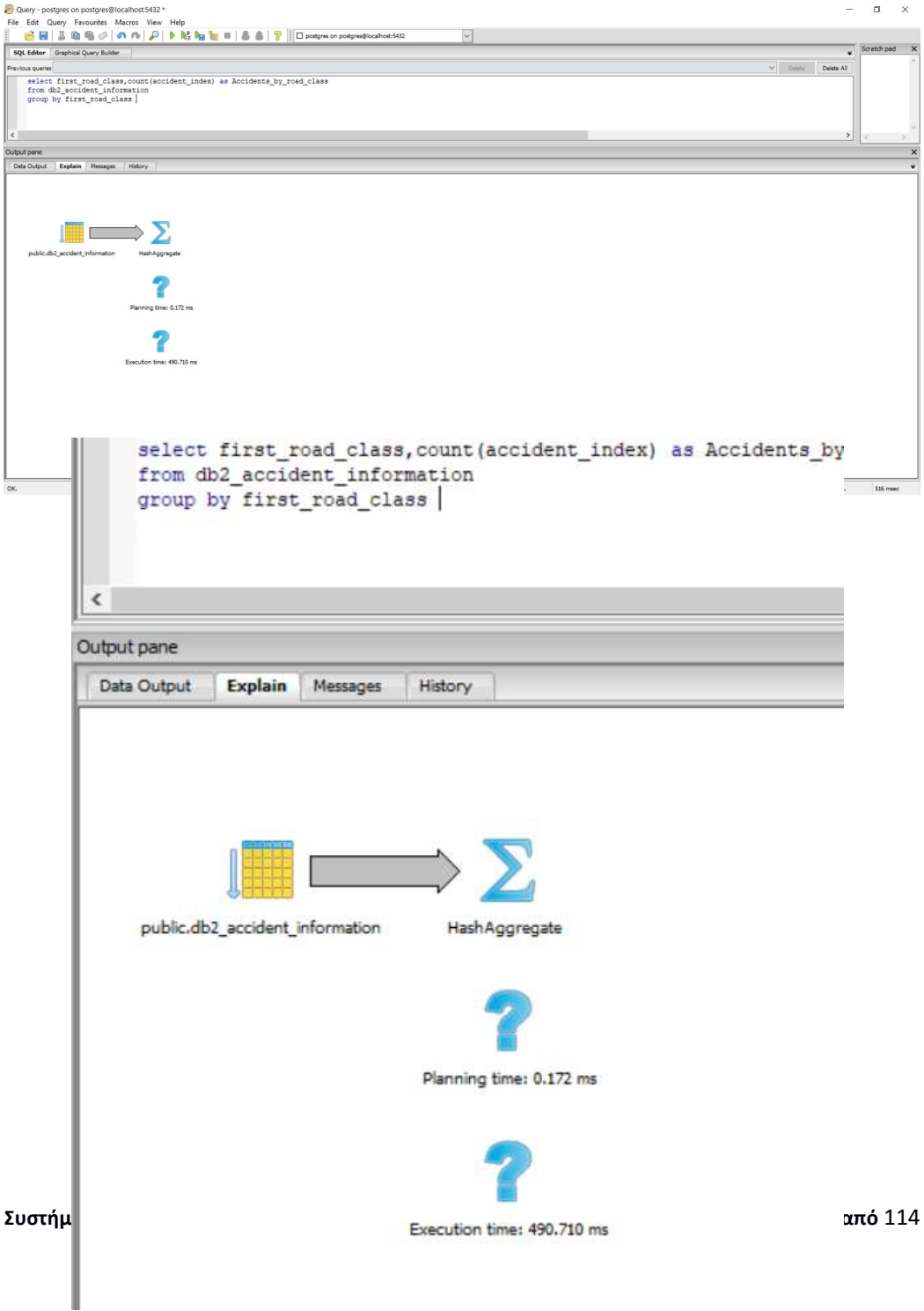
Στην συνέχεια αφού έχω κάνει restart τον server θα ξανά τρέξω τα sql queries μου

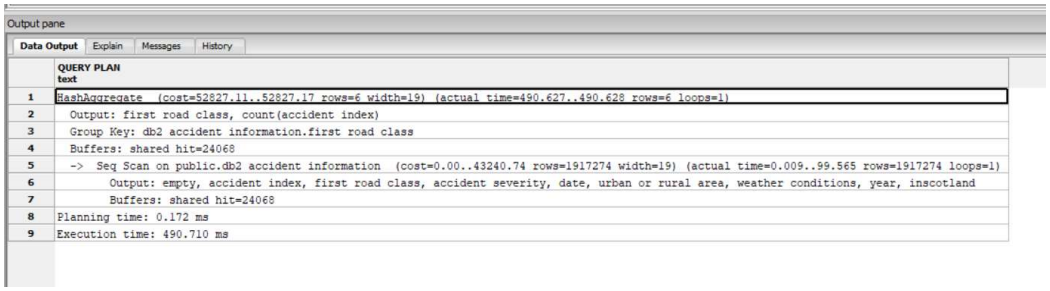
Σε κάθε εκτέλεση κάθε ερωτήματος έχω μια **1η εκτέλεση (αν και λέω πρώτη εκτέλεση μπορεί να έχω τρέξει περισσότερες από τρείς φορές την επερώτηση) από την εκφώνηση:**

(πάντα θα εκτελείτε την επερώτηση τουλάχιστον δύο φορές και θα κρατάτε τον τελευταίο χρόνο – ο λόγος που δεν κρατάμε απλά τον χρόνο εκτέλεσης της πρώτης φοράς είναι ότι δεν είναι αντιπροσωπευτικός γιατί τα buffers δεν έχουν προλάβει να αρχικοποιηθούν.

i. Βρείτε πόσα ατυχήματα έγιναν ανά κατηγορία δρόμου (road class).

1η Εκτέλεση(αυτή που μας ενδιαφέρει και κρατάμε(αν και λέω πρώτη εκτέλεση μπορεί να έχω τρέξει περισσότερες από τρείς φορές την επερώτηση))




Output pane

Data Output | Explain | Messages | History

	QUERY PLAN text
1	HashAggregate (cost=52827.11..52827.17 rows=6 width=19) (actual time=490.627..490.628 rows=6 loops=1)
2	Output: first road class, count(accident index)
3	Group Key: db2 accident information.first road class
4	Buffers: shared hit=24068
5	-> Seq Scan on public.db2 accident information (cost=0.00..43240.74 rows=1917274 width=19) (actual time=0.009..99.565 rows=1917274 loops=1)
6	Output: empty, accident index, first road class, accident severity, date, urban or rural area, weather conditions, year, inscotland
7	Buffers: shared hit=24068
8	Planning time: 0.172 ms
9	Execution time: 490.710 ms

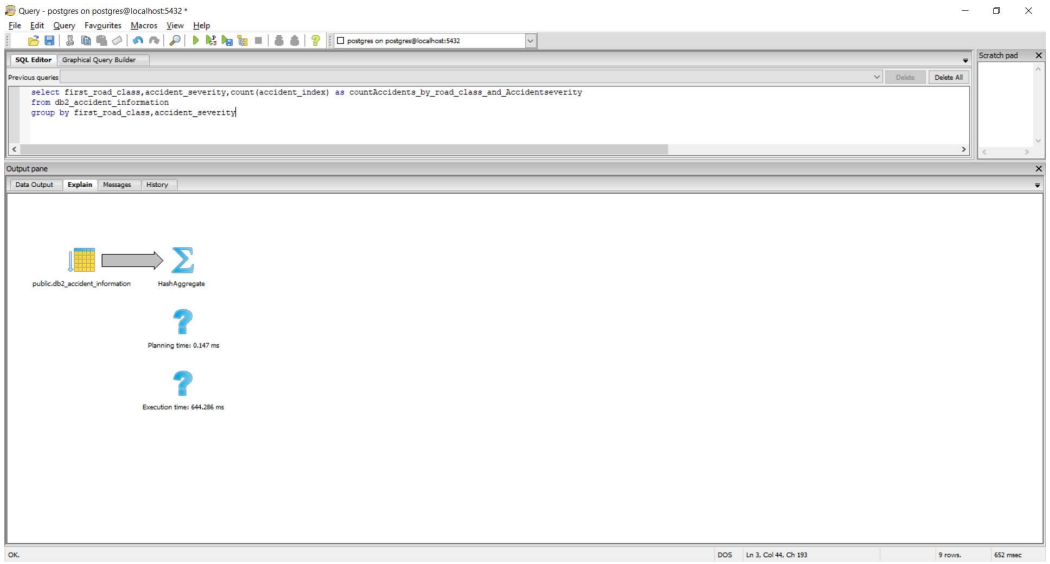
Planning time:0.172ms

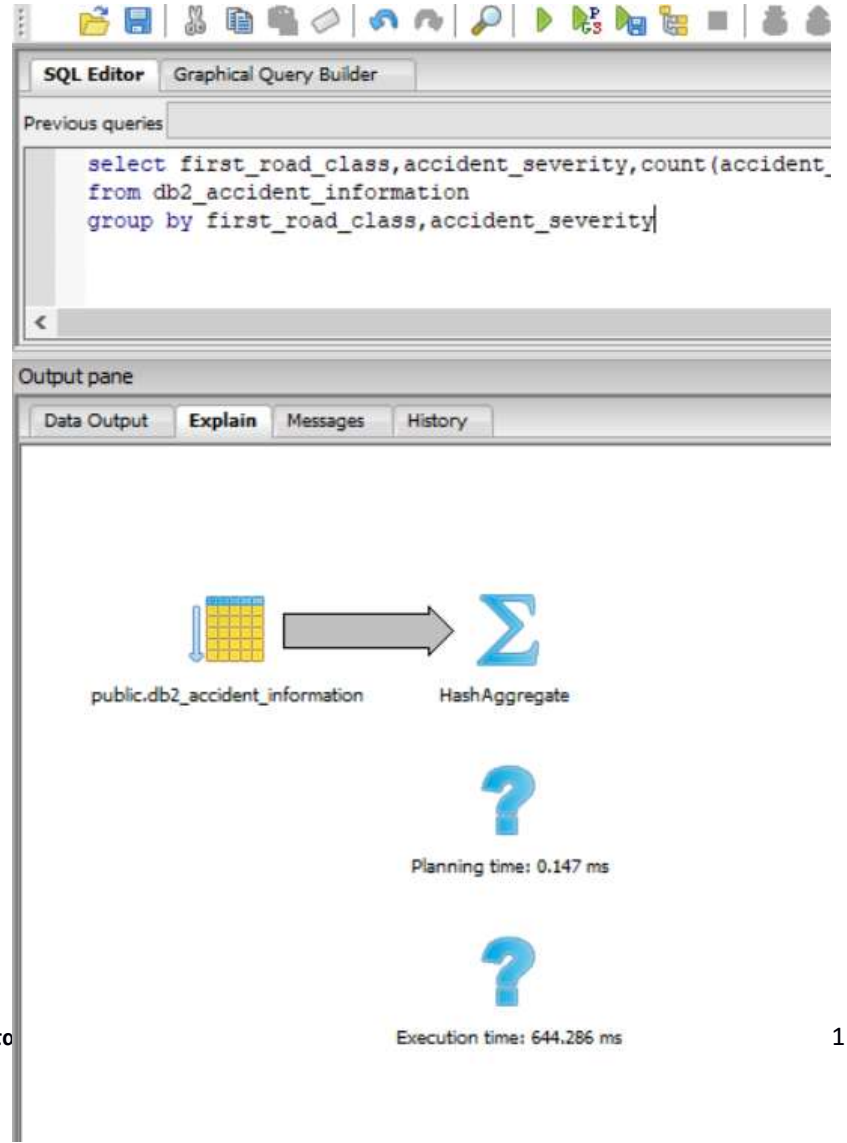
Execution time:490.710ms

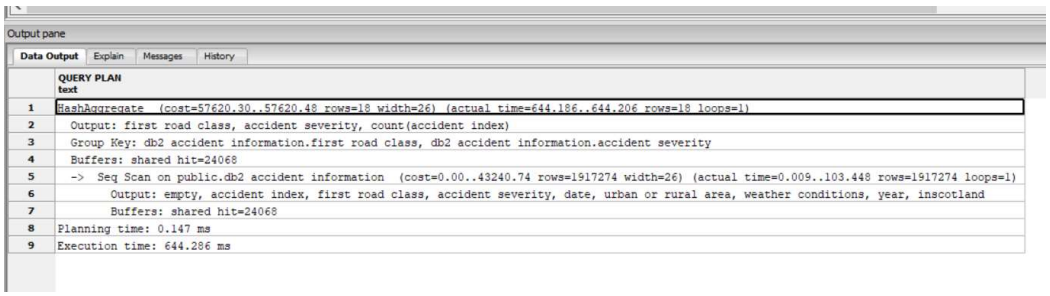
time:516ms

ii. Πόσα είναι τα ατυχήματα ανά κατηγορία δρόμου και ανά κατηγορία ατυχήματος (Accident Severity)

1η Εκτέλεση(αυτή που μας ενδιαφέρει και κρατάμε(αν και λέω πρώτη εκτέλεση μπορεί να έχω τρέξει περισσότερες από τρείς φορές την επερώτηση))


Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
SQL Editor
Graphical Query Builder
Previous queries
Scratch pad
Delete
Delete All
select first_road_class,accident_severity,count(accident_index) as countAccidents_by_road_class_and_Accidentseverity
from db2_accident_information
group by first_road_class,accident_severity
Output pane
Data Output
Explain
Messages
History
public.db2_accident_information
HashAggregate
Planning time: 0.147 ms
Execution time: 644.286 ms
OK.
DOS
Ln 3, Col 44, Ch 193
9 rows.
652 msec


SQL Editor
Graphical Query Builder
Previous queries
select first_road_class,accident_severity,count(accident_
from db2_accident_information
group by first_road_class,accident_severity
Output pane
Data Output
Explain
Messages
History
public.db2_accident_information
HashAggregate
Planning time: 0.147 ms
Execution time: 644.286 ms


Output pane

Data Output | Explain | Messages | History

	QUERY PLAN text
1	HashAggregate (cost=57620.30..57620.48 rows=18 width=26) (actual time=644.186..644.206 rows=18 loops=1)
2	Output: first road class, accident severity, count(accident index)
3	Group Key: db2 accident information.first road class, db2 accident information.accident severity
4	Buffers: shared hit=24068
5	-> Seq Scan on public.db2 accident information (cost=0.00..43240.74 rows=1917274 width=26) (actual time=0.009..103.448 rows=1917274 loops=1)
6	Output: empty, accident index, first road class, accident severity, date, urban or rural area, weather conditions, year, inscotland
7	Buffers: shared hit=24068
8	Planning time: 0.147 ms
9	Execution time: 644.286 ms

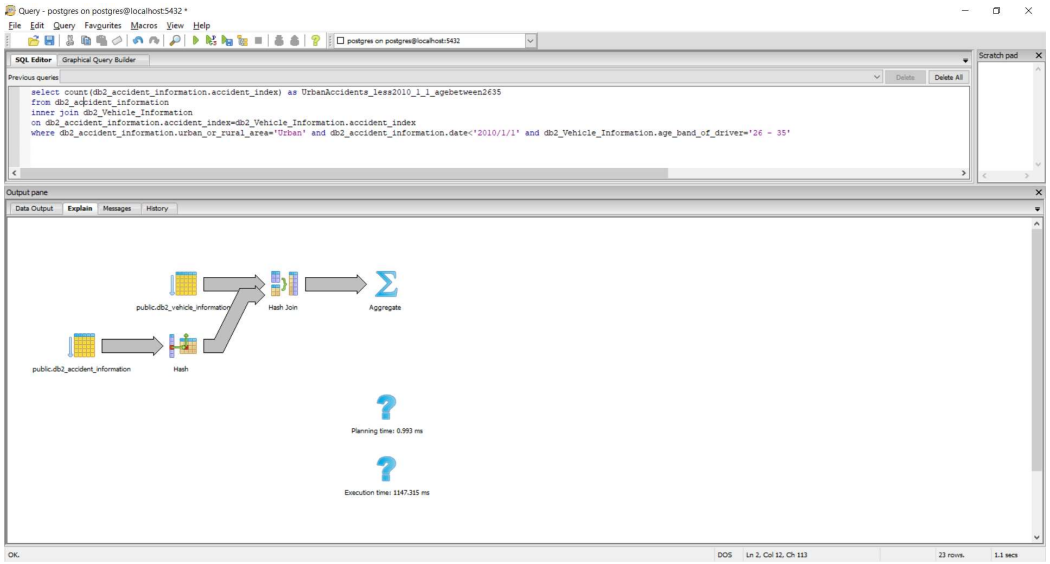
Planning time:0.147ms

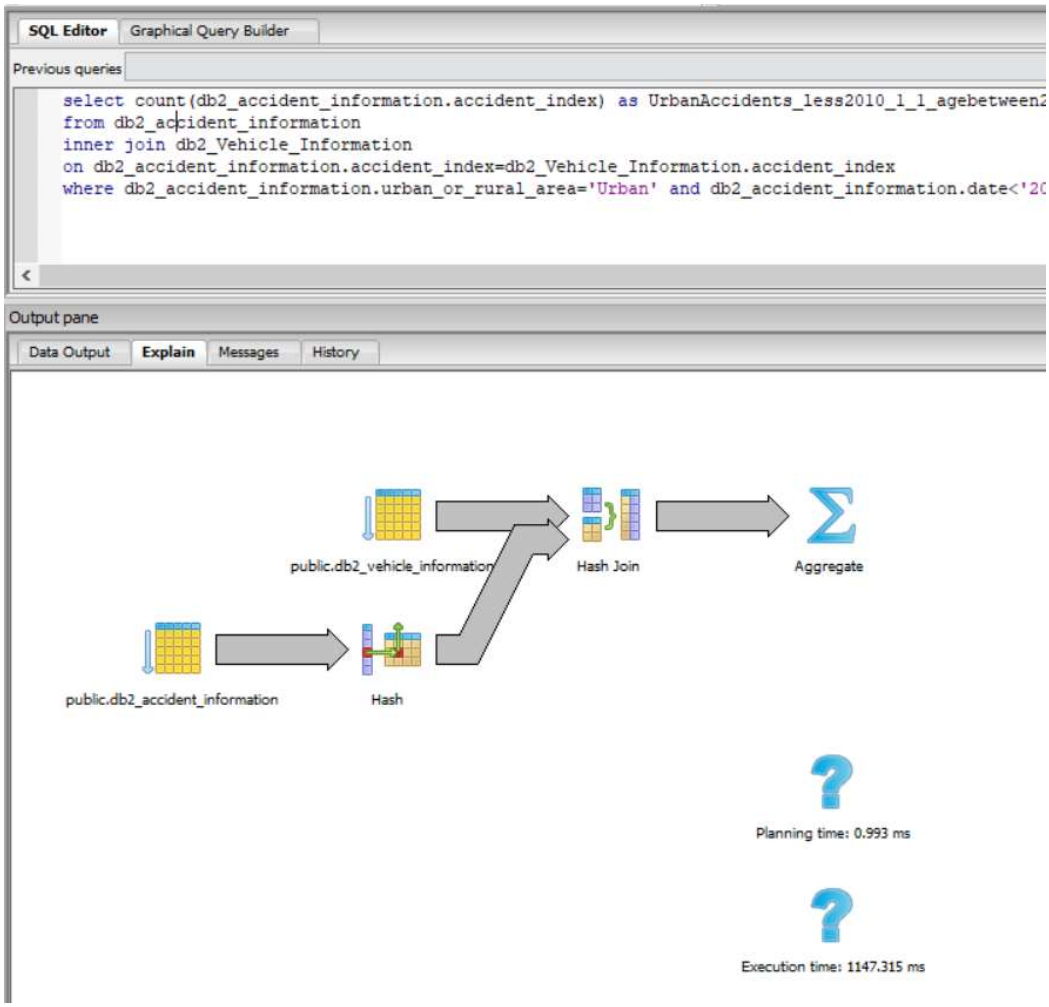
Execution time:644.286ms

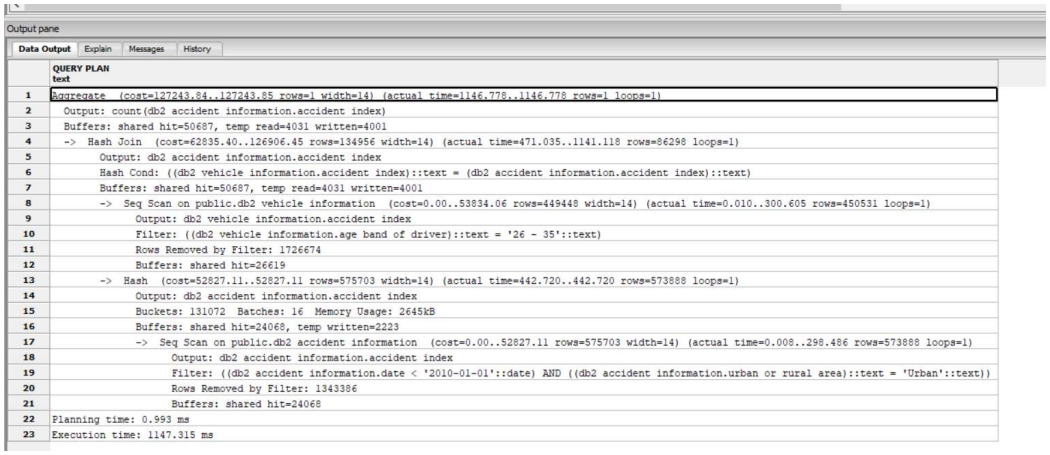
time:652ms

iii. Βρείτε πόσα ατυχήματα έγιναν σε αστική περιοχή πριν από την 1/1/2010 και η ηλικιακή κατηγορία του οδηγού είναι 26-35.

1η Εκτέλεση(αυτή που μας ενδιαφέρει και κρατάμε(αν και λέω πρώτη εκτέλεση μπορεί να έχω τρέξει περισσότερες από τρείς φορές την επερώτηση))


Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
SQL Editor
Graphical Query Builder
Previous queries
Delete
Delete All
Scratch pad
select count(db2_accident_information.accident_index) as UrbanAccidents_less2010_1_1_agebetween2635
from db2_accident_information
inner join db2_Vehicle_Information
on db2_accident_information.accident_index=db2_Vehicle_Information.accident_index
where db2_accident_information.urban_or_rural_area='Urban' and db2_accident_information.date<'2010/1/1' and db2_Vehicle_Information.age_band_of_driver='26 - 35'
Output pane
Data Output
Explain
Messages
History
public.db2_vehicle_information
Hash Join
Aggregate
public.db2_accident_information
Hash
Planning time: 0.993 ms
Execution time: 1147.315 ms
OK.
DOS
Ln 2, Col 12, Ch 113
23 rows.
1.1 secs


SQL Editor
Graphical Query Builder
Previous queries
select count(db2_accident_information.accident_index) as UrbanAccidents_less2010_1_1_agebetween2
from db2_accident_information
inner join db2_Vehicle_Information
on db2_accident_information.accident_index=db2_Vehicle_Information.accident_index
where db2_accident_information.urban_or_rural_area='Urban' and db2_accident_information.date<'20
Output pane
Data Output
Explain
Messages
History
public.db2_vehicle_information
Hash Join
Aggregate
public.db2_accident_information
Hash
Planning time: 0.993 ms
Execution time: 1147.315 ms



Output pane

Data Output | Explain | Messages | History

	QUERY PLAN text
1	Aggregate (cost=127243.84..127243.85 rows=1 width=14) (actual time=1146.778..1146.778 rows=1 loops=1)
2	Output: count(db2 accident information.accident index)
3	Buffers: shared hit=50687, temp read=4031 written=4001
4	-> Hash Join (cost=62835.40..126906.45 rows=134956 width=14) (actual time=471.035..1141.118 rows=86298 loops=1)
5	Output: db2 accident information.accident index
6	Hash Cond: ((db2 vehicle information.accident index)::text = (db2 accident information.accident index)::text)
7	Buffers: shared hit=50687, temp read=4031 written=4001
8	-> Seq Scan on public.db2 vehicle information (cost=0.00..53834.06 rows=449448 width=14) (actual time=0.010..300.605 rows=450531 loops=1)
9	Output: db2 vehicle information.accident index
10	Filter: ((db2 vehicle information.age band of driver)::text = '26 - 35'::text)
11	Rows Removed by Filter: 1726674
12	Buffers: shared hit=26619
13	-> Hash (cost=52827.11..52827.11 rows=575703 width=14) (actual time=442.720..442.720 rows=573888 loops=1)
14	Output: db2 accident information.accident index
15	Buckets: 131072 Batches: 16 Memory Usage: 2645kB
16	Buffers: shared hit=24068, temp written=2223
17	-> Seq Scan on public.db2 accident information (cost=0.00..52827.11 rows=575703 width=14) (actual time=0.008..298.486 rows=573888 loops=1)
18	Output: db2 accident information.accident index
19	Filter: ((db2 accident information.date < '2010-01-01'::date) AND ((db2 accident information.urban or rural area)::text = 'Urban'::text))
20	Rows Removed by Filter: 1343386
21	Buffers: shared hit=24068
22	Planning time: 0.993 ms
23	Execution time: 1147.315 ms

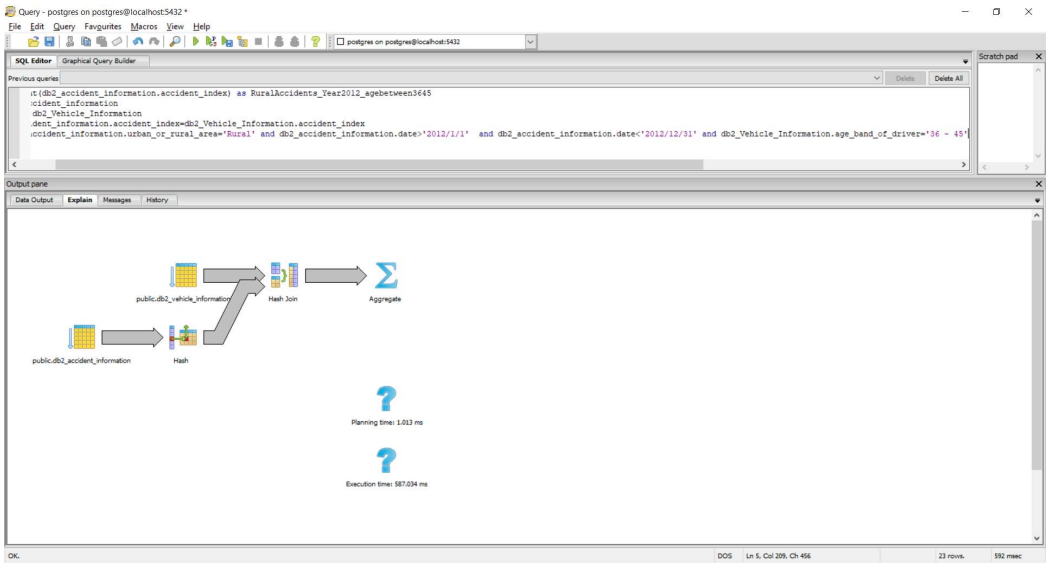
Planning time:0.993ms

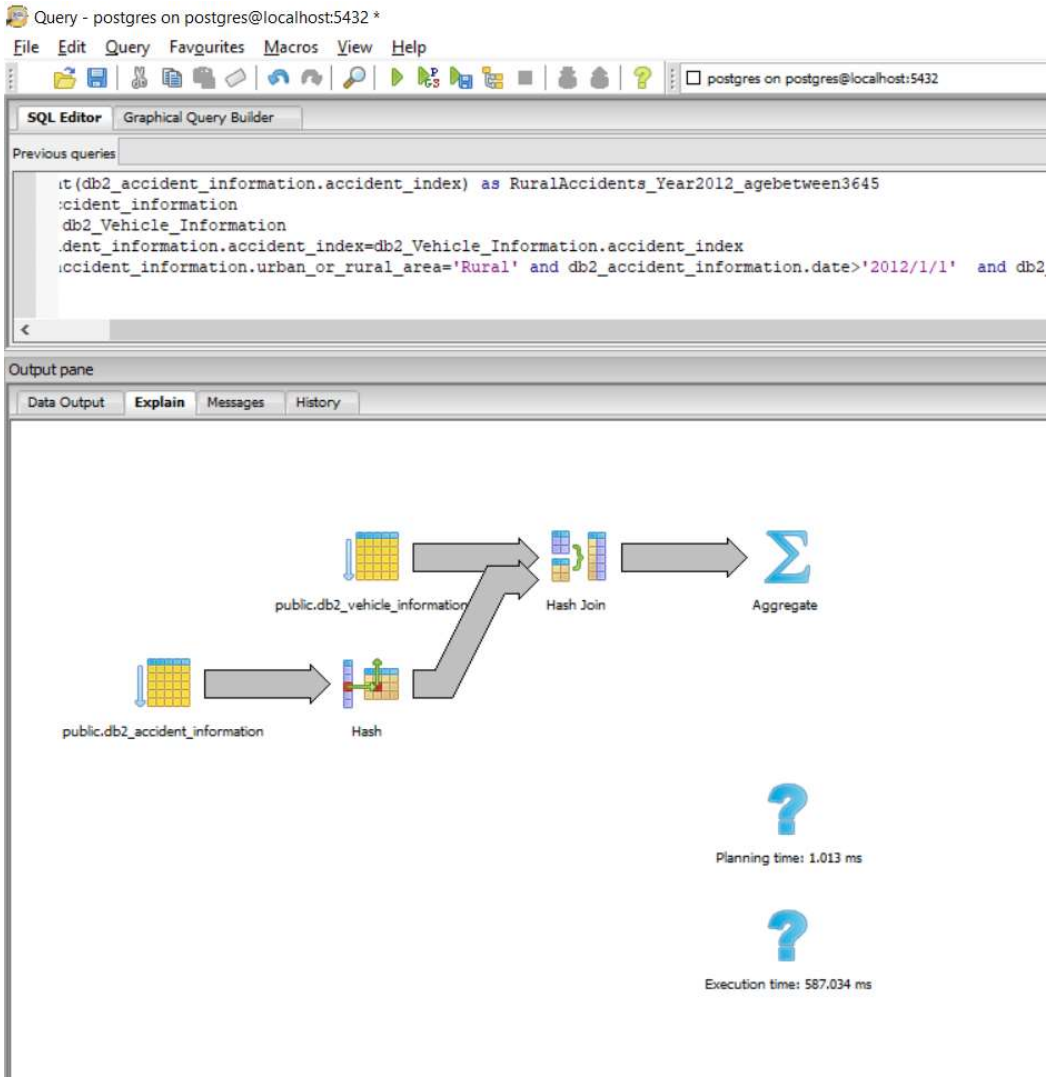
Execution time:1147.315ms

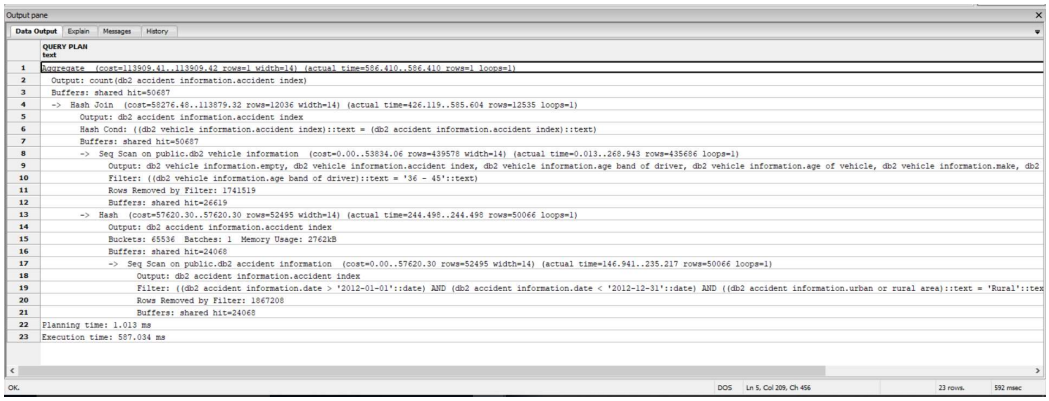
time:1.1ms

iv. Βρείτε πόσα ατυχήματα έγιναν σε αγροτική περιοχή το έτος 2012 και η ηλικιακή κατηγορία του οδηγού είναι 36-45.

1η Εκτέλεση(αυτή που μας ενδιαφέρει και κρατάμε(αν και λέω πρώτη εκτέλεση μπορεί να έχω τρέξει περισσότερες από τρείς φορές την επερώτηση))




Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
postgres on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
t(db2_accident_information.accident_index) as RuralAccidents_Year2012_agebetween3645
cident_information
db2_Vehicle_Information
dent_information.accident_index=db2_Vehicle_Information.accident_index
ccident_information.urban_or_rural_area='Rural' and db2_accident_information.date>'2012/1/1' and db2
Output pane
Data Output
Explain
Messages
History
public.db2_vehicle_information
Hash Join
Aggregate
public.db2_accident_information
Hash
Planning time: 1.013 ms
Execution time: 587.034 ms



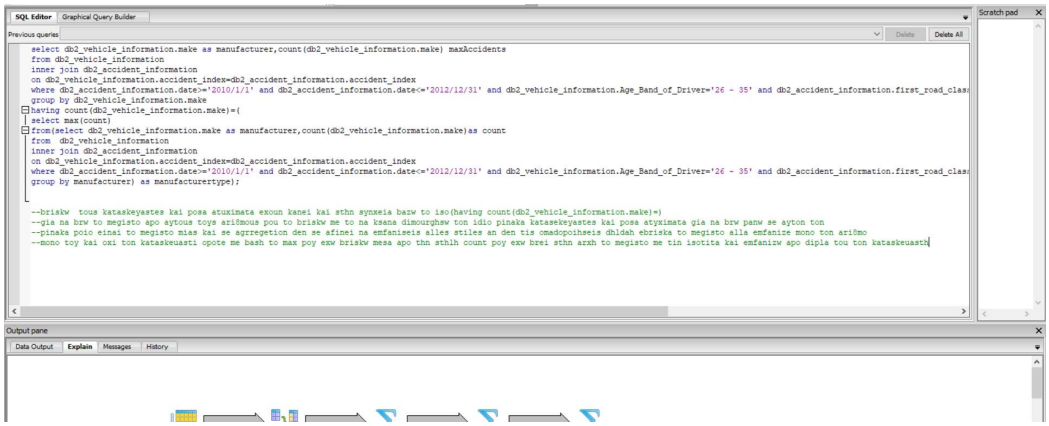
Planning time:1.013ms

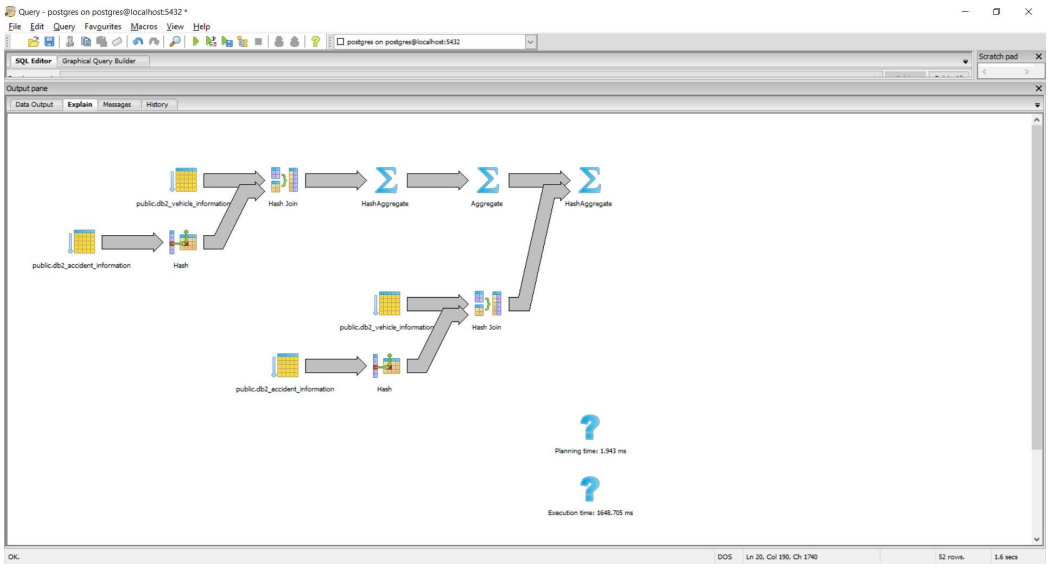
Execution time:587.034ms

time:592msec

v. Βρείτε τον κατασκευαστή του οχήματος που έχει τα περισσότερα ατυχήματα μεταξύ των ετών 2010 και 2012, η ηλικιακή κατηγορία των οδηγών είναι 26- 35 και η κατηγορία δρόμου είναι Α.

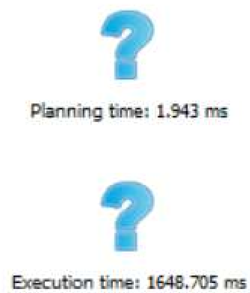
1η Εκτέλεση(αυτή που μας ενδιαφέρει και κρατάμε(αν και λέω πρώτη εκτέλεση μπορεί να έχω τρέξει περισσότερες από τρείς φορές την επερώτηση))

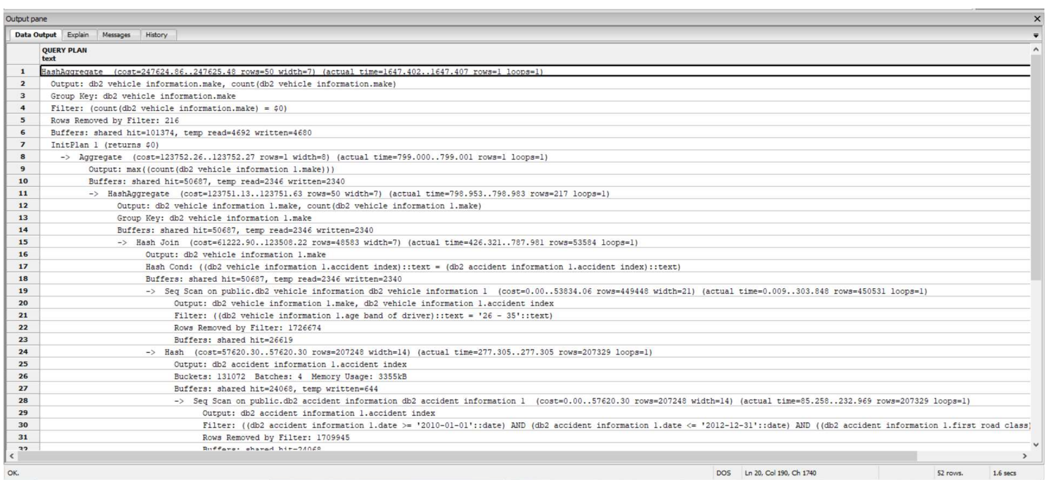




Planning time:1.943ms

Execution time:1648.705



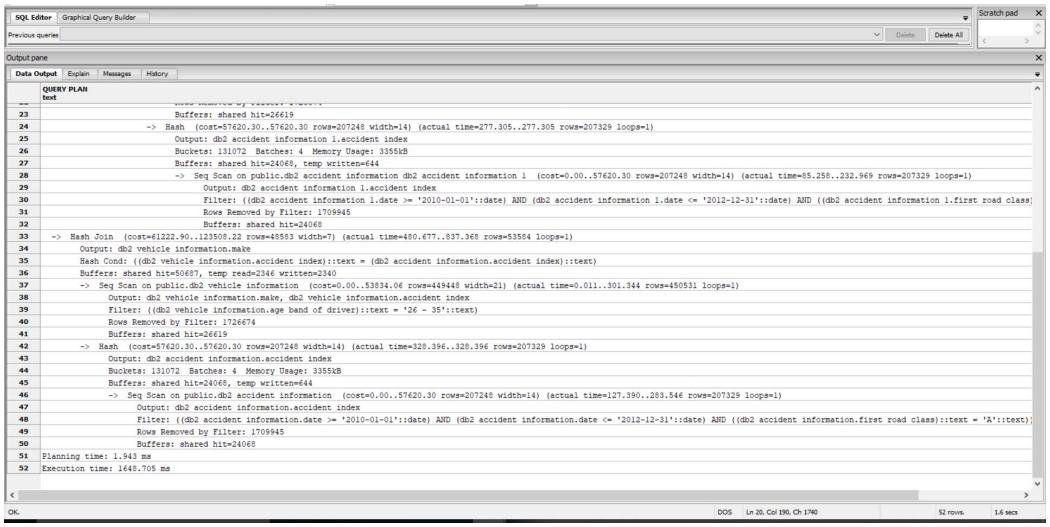


Output pane

Data Output | Explain | Messages | History

	QUERY PLAN text
1	HashAggregate (cost=247624.86..247625.48 rows=50 width=7) (actual time=1647.402..1647.407 rows=1 loops=1)
2	Output: db2 vehicle information.make, count(db2 vehicle information.make)
3	Group Key: db2 vehicle information.make
4	Filter: (count(db2 vehicle information.make) = $0)
5	Rows Removed by Filter: 216
6	Buffers: shared hit=101374, temp read=4692 written=4680
7	InitPlan 1 (returns $0)
8	-> Aggregate (cost=123752.26..123752.27 rows=1 width=8) (actual time=799.000..799.001 rows=1 loops=1)
9	Output: max((count(db2 vehicle information 1.make)))
10	Buffers: shared hit=50687, temp read=2346 written=2340
11	-> HashAggregate (cost=123751.13..123751.63 rows=50 width=7) (actual time=798.953..798.983 rows=217 loops=1)
12	Output: db2 vehicle information 1.make, count(db2 vehicle information 1.make)
13	Group Key: db2 vehicle information 1.make
14	Buffers: shared hit=50687, temp read=2346 written=2340
15	-> Hash Join (cost=61222.90..123508.22 rows=48583 width=7) (actual time=426.321..787.981 rows=53584 loops=1)
16	Output: db2 vehicle information 1.make
17	Hash Cond: ((db2 vehicle information 1.accident index)::text = (db2 accident information 1.accident index)::text)
18	Buffers: shared hit=50687, temp read=2346 written=2340
19	-> Seq Scan on public.db2 vehicle information db2 vehicle information 1 (cost=0.00..53834.06 rows=449448 width=21) (actual time=0.009..303.848 rows=450531 loops=1)
20	Output: db2 vehicle information 1.make, db2 vehicle information 1.accident index
21	Filter: ((db2 vehicle information 1.age band of driver)::text = '26 - 35'::text)
22	Rows Removed by Filter: 1726674
23	Buffers: shared hit=26619
24	-> Hash (cost=57620.30..57620.30 rows=207248 width=14) (actual time=277.305..277.305 rows=207329 loops=1)
25	Output: db2 accident information 1.accident index
26	Buckets: 131072 Batches: 4 Memory Usage: 3355kB
27	Buffers: shared hit=24068, temp written=644
28	-> Seq Scan on public.db2 accident information db2 accident information 1 (cost=0.00..57620.30 rows=207248 width=14) (actual time=85.258..232.969 rows=207329 loops=1)
29	Output: db2 accident information 1.accident index
30	Filter: ((db2 accident information 1.date >= '2010-01-01'::date) AND (db2 accident information 1.date <= '2012-12-31'::date) AND ((db2 accident information 1.first road class)
31	Rows Removed by Filter: 1709945

OK. DOS Ln 20, Col 190, Ch 1740 52 rows. 1.6 secs



Planning time:1.943ms

Execution time:1648.705

time:1.6secs

Καταγράφουμε τώρα τα στοιχεία σε πίνακες

1ο Πίνακας(όλα είναι by default δεν έχουμε πειράξει τίποτα)

Query	Planning time	Execution time	Total time
i	0.061ms	449.906ms	469msecs
ii	0.147ms	620.265ms	641msecs
iii	0.977ms	1061.916ms	1secs
iv	0.983ms	557.107ms	578msecs
v	0.709ms	1574.724ms	1.5secs

Shared buffers:16GB

Query	Planning time	Execution time	Total time

i	0.172ms	490.710ms	516msecs
ii	0.147ms	644.286ms	652msecs
iii	0.993ms	1147.315ms	1.1secs
iv	1.013ms	587.034ms	592msecs
v	1.943ms	1648.705ms	1.6secs

Δυο πινακες μαζί

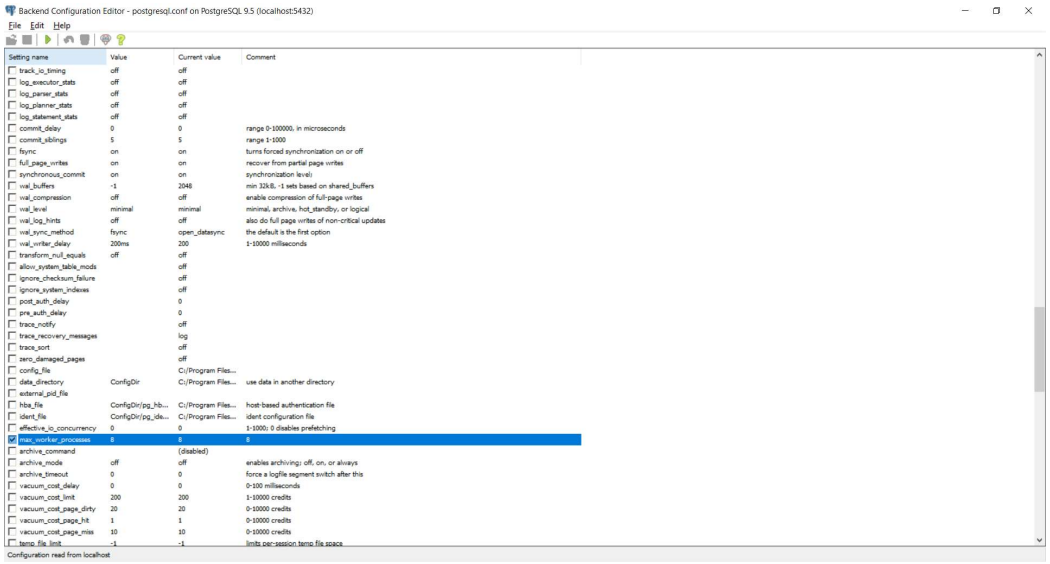
Query	Planning time	**Planning time share buffers 16GB**	Execution time	**Execution time share buffers 16GB**	Total time	**Total time share buffers 16GB**
i	0.061ms	**0.172ms**	449.906ms	**490.710ms**	469msecs	**516msecs**
ii	0.147ms	**0.147ms**	620.265ms	**644.286ms**	641msecs	**652msecs**
iii	0.977ms	**0.993ms**	1061.916ms	**1147.315ms**	1secs	**1.1secs**
iv	0.983ms	**1.013ms**	557.107ms	**587.034ms**	578msecs	**592msecs**
v	0.709ms	**1.943ms**	1574.724ms	**1648.705ms**	1.5secs	**1.6secs**

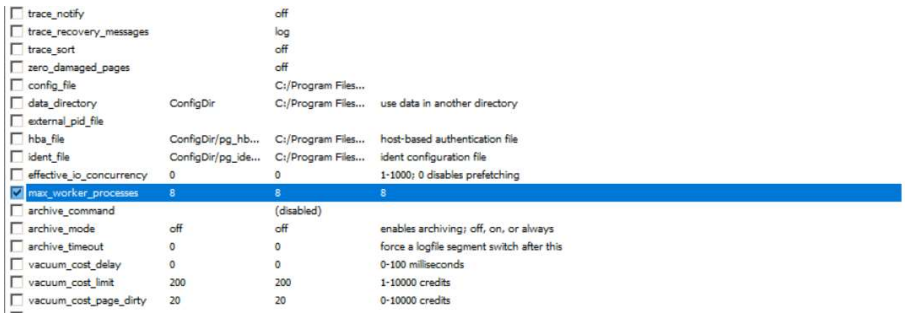
Όπως παρατηρούμε επειδή έχουμε αυξήσει το μέγεθος της μνήμης τόσο ώστε να είναι μεγαλύτερο από την μνήμη ram στα 16GB παρατηρούμε μια αύξηση του χρόνου εκτέλεσης των επερωτήσεων και αυτο είναι φυσιολογικό αφού δεν κάνει όλους τους υπολογισμούς μέσα στην ram.

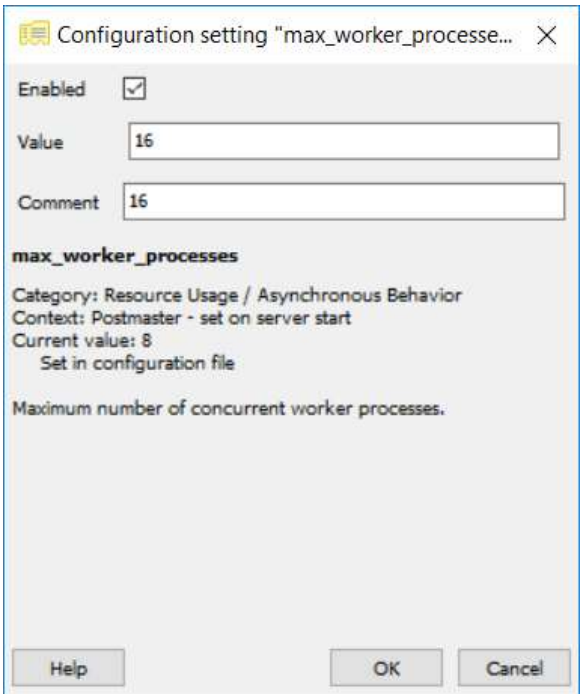
Στην συνέχεια για όλες τις υπόλοιπες εκτελέσεις(μπορεί να έχω τρέξει περισσότερες από τρείς φορές τις επερωτήσεις κάθε φορά απλά δεν το γράφω κάθε φορά που τρέχω ένα query))

(c) Ρυθμίστε την PostgreSQL έτσι ώστε να χρησιμοποιεί όλη την επεξεργαστική ισχύ του υπολογιστή σας. Έπειτα, εκτελέστε πάλι τις παραπάνω επερωτήσεις και εξηγήστε τι παρατηρείτε. TIP: max_parallel_workers_per_gather

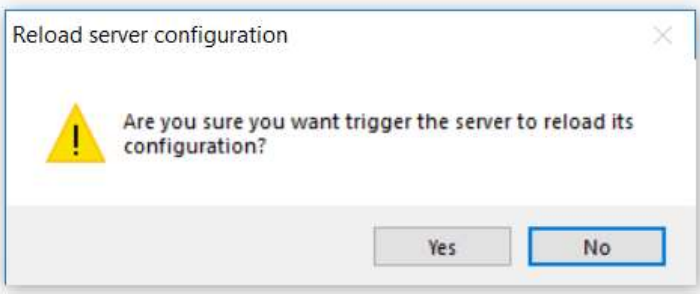
Παω εκει που λέει tools→Server Configuration → postgresql.conf


Backend Configuration Editor - postgresql.conf on PostgreSQL 9.5 (localhost:5432)
File Edit Help
Setting name Value Current value Comment
track_io_timing off off
log_executor_stats off off
log_parser_stats off off
log_planner_stats off off
log_statement_stats off off
commit_delay 0 0 range 0-100000, in microseconds
commit_siblings 5 5 range 1-1000
fsync on on turns forced synchronization on or off
full_page_writes on on recover from partial page writes
synchronous_commit on on synchronization level
wal_buffers -1 2048 min 32kB, -1 sets based on shared_buffers
wal_compression off off enable compression of full-page writes
wal_level minimal minimal minimal, archive, hot_standby, or logical
wal_log_hints off off also do full page writes of non-critical updates
wal_sync_method fsync open_datasync the default is the first option
wal_writer_delay 200ms 200 1-10000 milliseconds
transform_null_equals off off
allow_system_table_mods off
ignore_checksum_failure off
ignore_system_indexes off
post_auth_delay 0
pre_auth_delay 0
trace_notify off
trace_recovery_messages log
trace_sort off
zero_damaged_pages off
config_file C:/Program Files...
data_directory ConfigDir C:/Program Files... use data in another directory
external_pid_file
hba_file ConfigDir/pg_hb... C:/Program Files... host-based authentication file
ident_file ConfigDir/pg_ide... C:/Program Files... ident configuration file
effective_io_concurrency 0 0 1-1000; 0 disables prefetching
max_worker_processes 8 8 8
archive_command (disabled)
archive_mode off off enables archiving; off, on, or always
archive_timeout 0 0 force a logfile segment switch after this
vacuum_cost_delay 0 0 0-100 milliseconds
vacuum_cost_limit 200 200 1-10000 credits
vacuum_cost_page_dirty 20 20 0-10000 credits
vacuum_cost_page_hit 1 1 0-10000 credits
vacuum_cost_page_miss 10 10 0-10000 credits
temp_file_limit -1 -1 limits per-session temp file space
Configuration read from localhost


trace_notify off
trace_recovery_messages log
trace_sort off
zero_damaged_pages off
config_file C:/Program Files...
data_directory ConfigDir C:/Program Files... use data in another directory
external_pid_file
hba_file ConfigDir/pg_hb... C:/Program Files... host-based authentication file
ident_file ConfigDir/pg_ide... C:/Program Files... ident configuration file
effective_io_concurrency 0 0 1-1000; 0 disables prefetching
max_worker_processes 8 8 8
archive_command (disabled)
archive_mode off off enables archiving; off, on, or always
archive_timeout 0 0 force a logfile segment switch after this
vacuum_cost_delay 0 0 0-100 milliseconds
vacuum_cost_limit 200 200 1-10000 credits
vacuum_cost_page_dirty 20 20 0-10000 credits

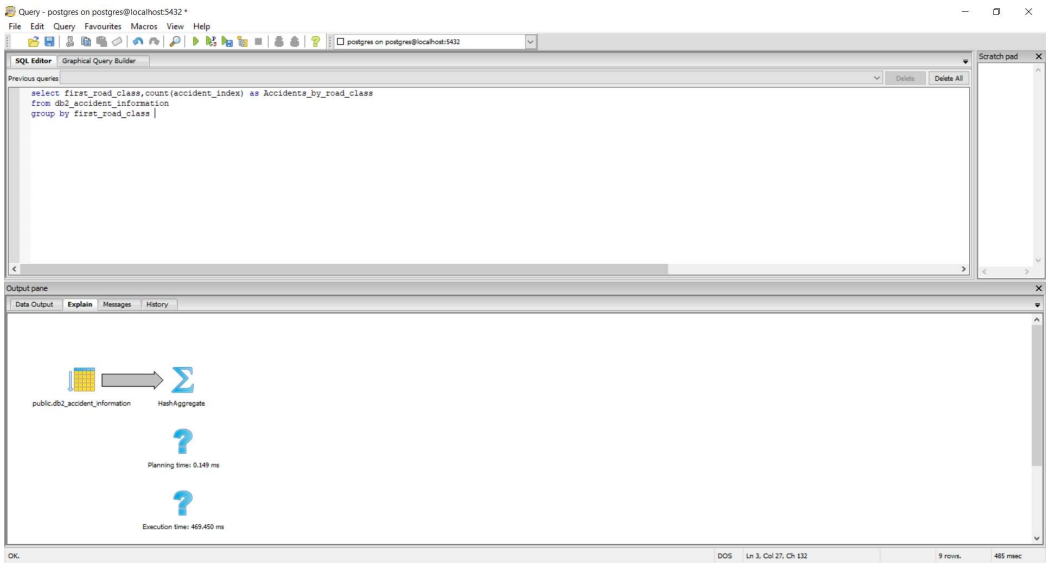


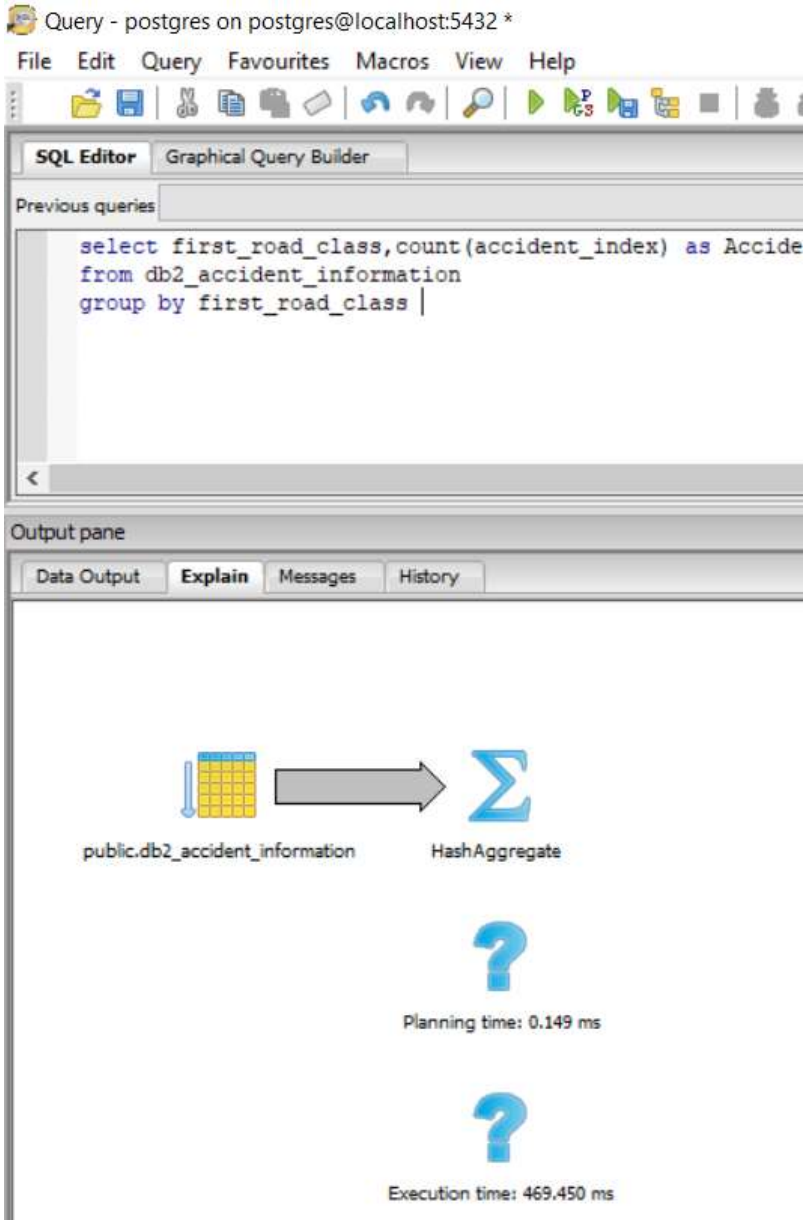
Θέτουμε τους maxworkers 16

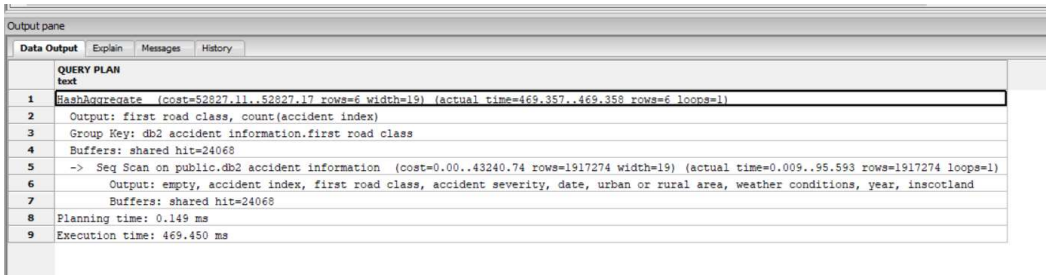


data_directory	ConfigDir	C:/Program Files...	use data in another directory
external_pid_file			
hba_file	ConfigDir/pg_hb...	C:/Program Files...	host-based authentication file
ident_file	ConfigDir/pg_ide...	C:/Program Files...	ident configuration file
effective_io_concurrency	0	0	1-1000; 0 disables prefetching
max_worker_processes	16	8	16
archive_command		(disabled)	
archive_mode	off	off	enables archiving; off, on, or always
archive_timeout	0	0	force a logfile segment switch after this
vacuum_cost_delay	0	0	0-100 milliseconds

i. Βρείτε πόσα ατυχήματα έγιναν ανά κατηγορία δρόμου (road class).




Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
SQL Editor
Graphical Query Builder
Previous queries
select first_road_class,count(accident_index) as Accide
from db2_accident_information
group by first_road_class
Output pane
Data Output
Explain
Messages
History
public.db2_accident_information
HashAggregate
Planning time: 0.149 ms
Execution time: 469.450 ms

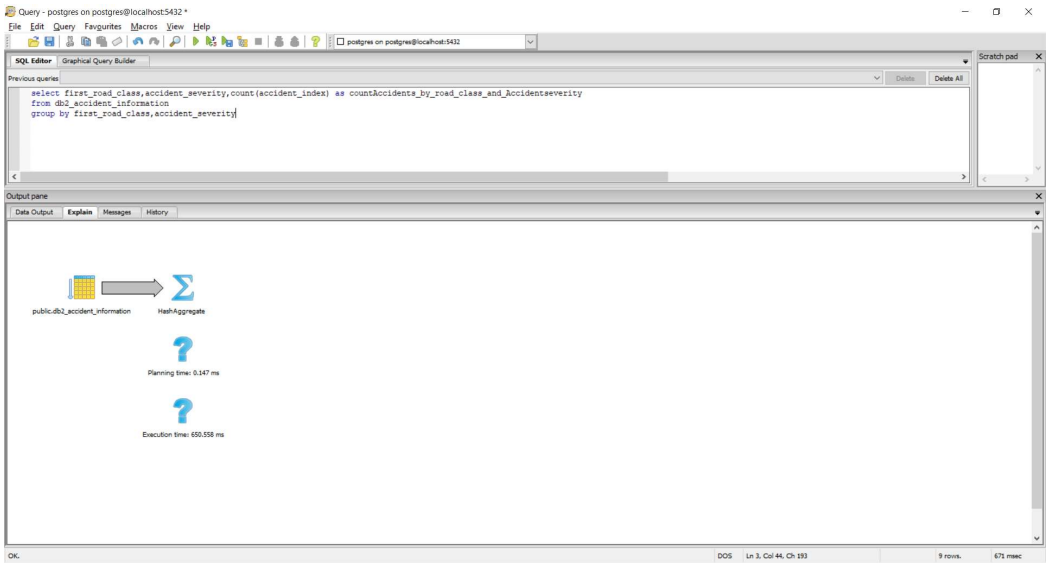


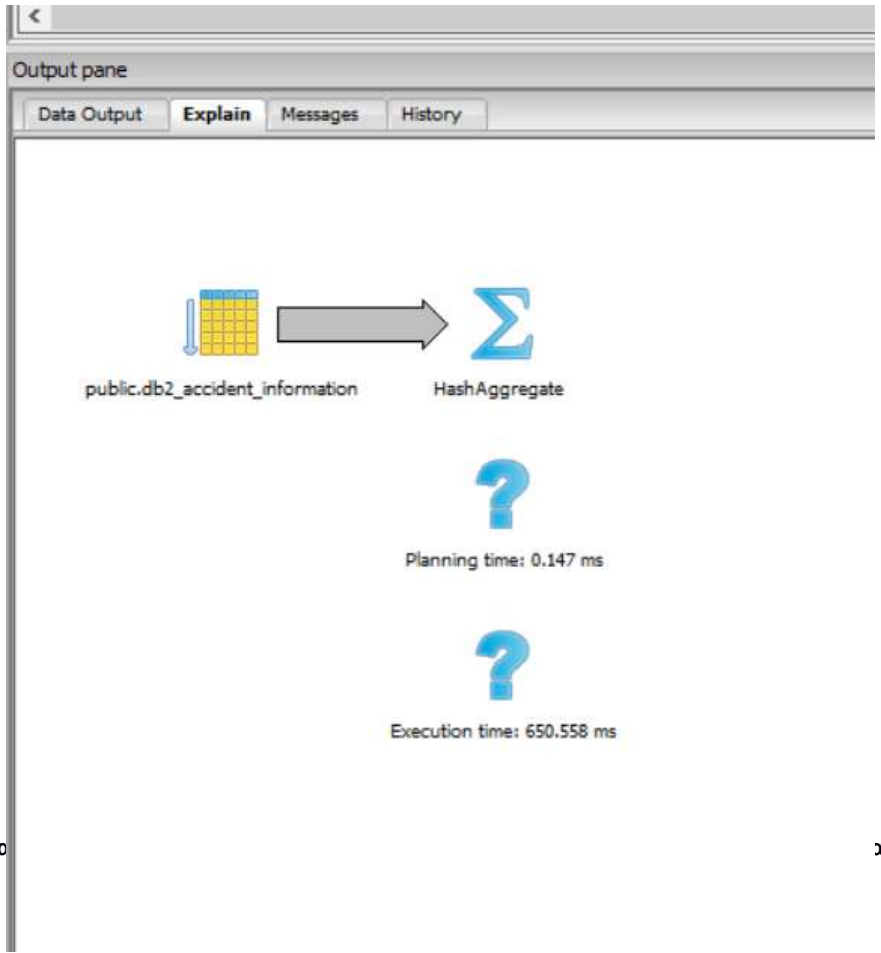
Planning time 0.149ms

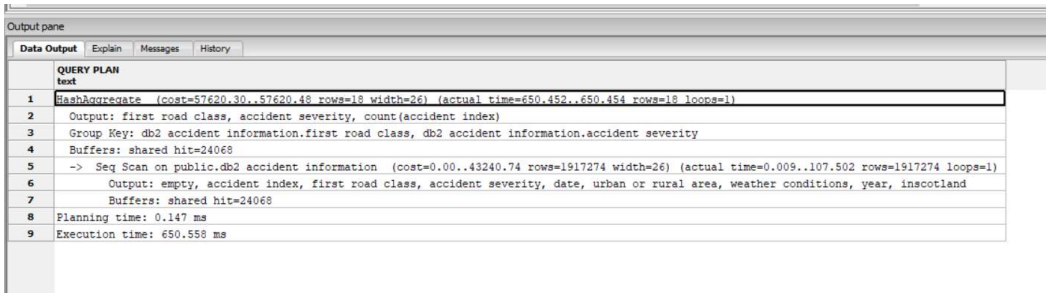
Execution time 469.450ms

time 485msecs

ii. Πόσα είναι τα ατυχήματα ανά κατηγορία δρόμου και ανά κατηγορία ατυχήματος (Accident Severity)




Output pane
Data Output
Explain
Messages
History
public.db2_accident_information
HashAggregate
Planning time: 0.147 ms
Execution time: 650.558 ms


Output pane

Data Output | Explain | Messages | History

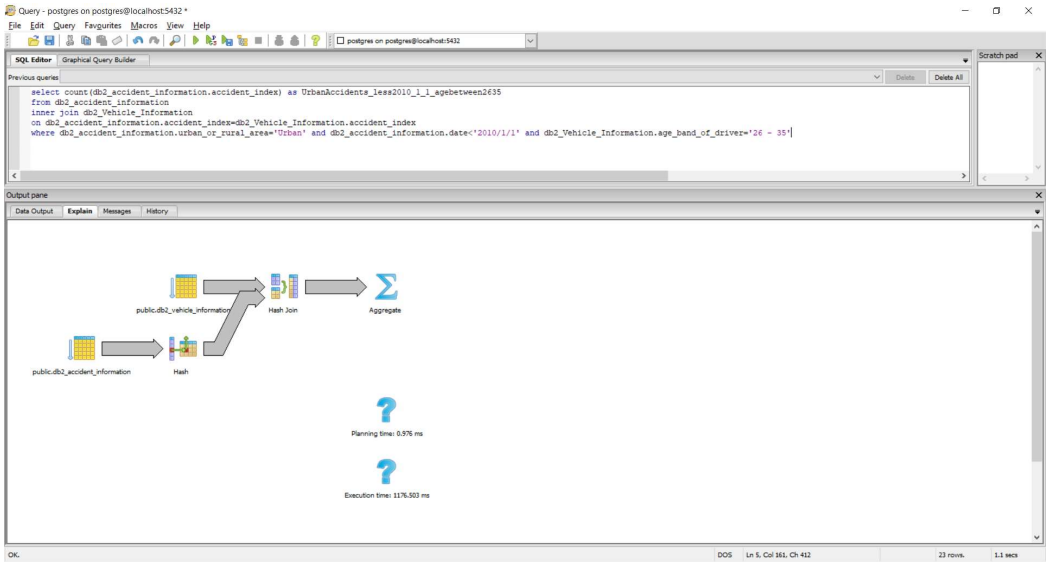
	QUERY PLAN text
1	HashAggregate (cost=57620.30..57620.48 rows=18 width=26) (actual time=650.452..650.454 rows=18 loops=1)
2	Output: first road class, accident severity, count(accident index)
3	Group Key: db2 accident information.first road class, db2 accident information.accident severity
4	Buffers: shared hit=24068
5	-> Seq Scan on public.db2 accident information (cost=0.00..43240.74 rows=1917274 width=26) (actual time=0.009..107.502 rows=1917274 loops=1)
6	Output: empty, accident index, first road class, accident severity, date, urban or rural area, weather conditions, year, inscotland
7	Buffers: shared hit=24068
8	Planning time: 0.147 ms
9	Execution time: 650.558 ms

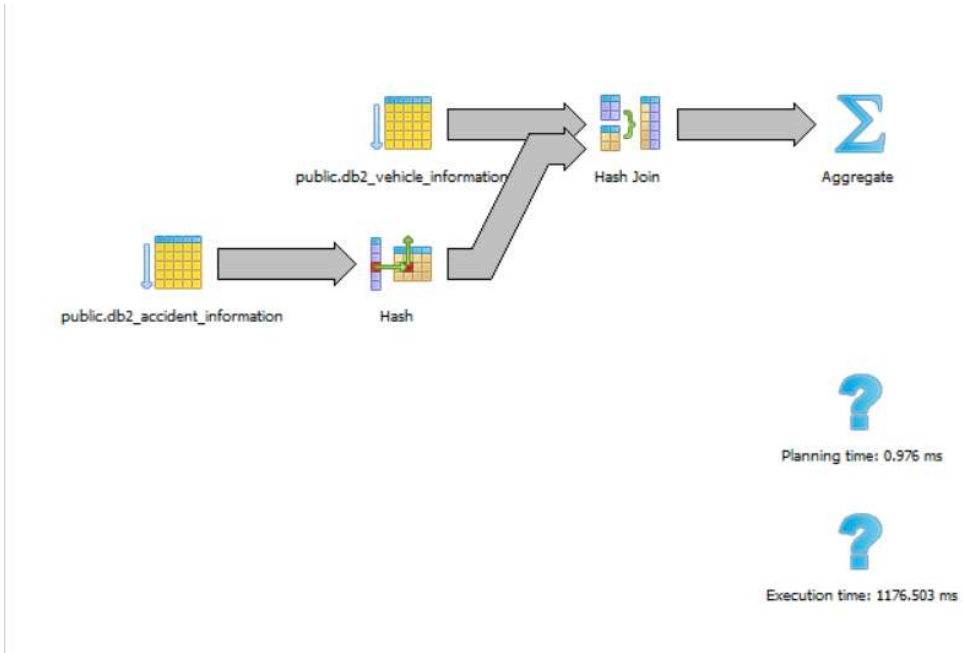
Planning time 0.147ms

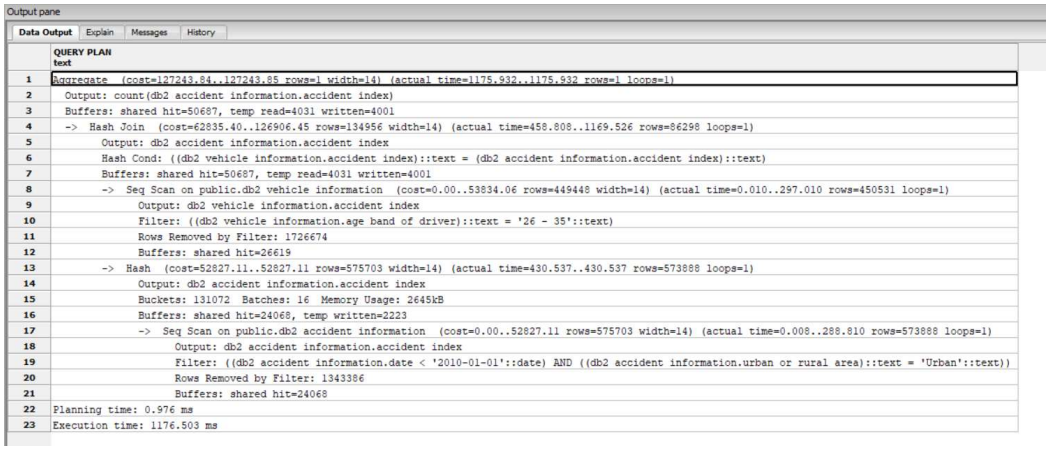
Execution time 450.558ms

time 671msecs

iii. Βρείτε πόσα ατυχήματα έγιναν σε αστική περιοχή πριν από την 1/1/2010 και η ηλικιακή κατηγορία του οδηγού είναι 26-35.


Query - postgres on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
SQL Editor
Graphical Query Builder
Previous queries
Scratch pad
Delete
Delete All
select count(db2_accident_information.accident_index) as UrbanAccidents_less2010_1_1_agebetween2635
from db2_accident_information
inner join db2_Vehicle_Information
on db2_accident_information.accident_index=db2_Vehicle_Information.accident_index
where db2_accident_information.urban_or_rural_area='Urban' and db2_accident_information.date<'2010/1/1' and db2_Vehicle_Information.age_band_of_driver='26 - 35'
Output pane
Data Output
Explain
Messages
History
public.db2_vehicle_information
Hash Join
Aggregate
public.db2_accident_information
Hash
Planning time: 0.976 ms
Execution time: 1176.503 ms
OK.
DOS
Ln 5, Col 161, Ch 412
23 rows.
1.1 secs


public.db2_vehicle_information
Hash Join
Aggregate
public.db2_accident_information
Hash
Planning time: 0.976 ms
Execution time: 1176.503 ms



Output pane

Data Output | Explain | Messages | History

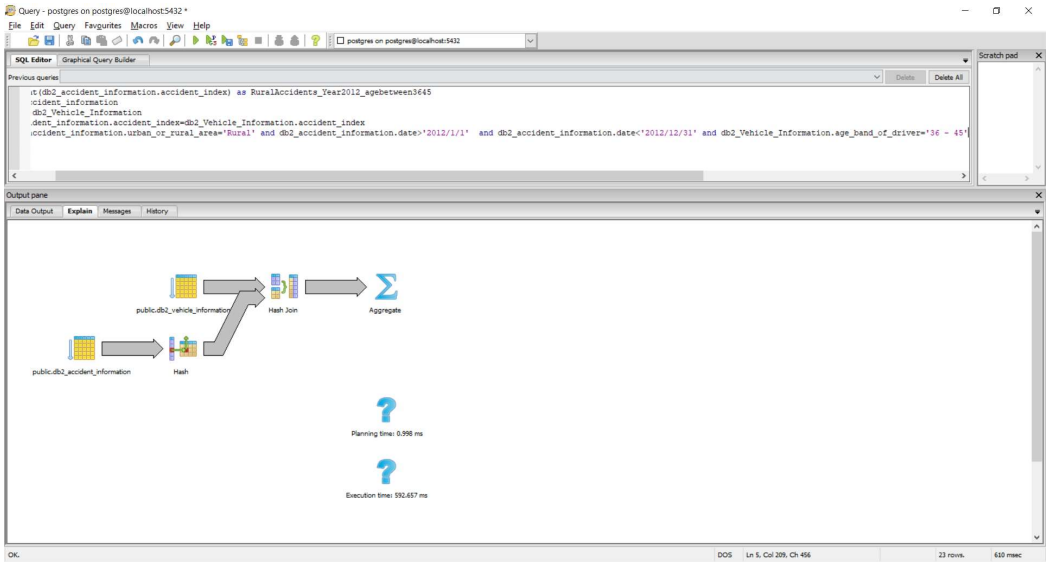
	QUERY PLAN text
1	Aggregate (cost=127243.84..127243.85 rows=1 width=14) (actual time=1175.932..1175.932 rows=1 loops=1)
2	Output: count(db2 accident information.accident index)
3	Buffers: shared hit=50687, temp read=4031 written=4001
4	-> Hash Join (cost=62835.40..126906.45 rows=134956 width=14) (actual time=458.808..1169.526 rows=86298 loops=1)
5	Output: db2 accident information.accident index
6	Hash Cond: ((db2 vehicle information.accident index)::text = (db2 accident information.accident index)::text)
7	Buffers: shared hit=50687, temp read=4031 written=4001
8	-> Seq Scan on public.db2 vehicle information (cost=0.00..53834.06 rows=449448 width=14) (actual time=0.010..297.010 rows=450531 loops=1)
9	Output: db2 vehicle information.accident index
10	Filter: ((db2 vehicle information.age band of driver)::text = '26 - 35'::text)
11	Rows Removed by Filter: 1726674
12	Buffers: shared hit=26619
13	-> Hash (cost=52827.11..52827.11 rows=575703 width=14) (actual time=430.537..430.537 rows=573888 loops=1)
14	Output: db2 accident information.accident index
15	Buckets: 131072 Batches: 16 Memory Usage: 2645kB
16	Buffers: shared hit=24068, temp written=2223
17	-> Seq Scan on public.db2 accident information (cost=0.00..52827.11 rows=575703 width=14) (actual time=0.008..288.810 rows=573888 loops=1)
18	Output: db2 accident information.accident index
19	Filter: ((db2 accident information.date < '2010-01-01'::date) AND ((db2 accident information.urban or rural area)::text = 'Urban'::text))
20	Rows Removed by Filter: 1343386
21	Buffers: shared hit=24068
22	Planning time: 0.976 ms
23	Execution time: 1176.503 ms

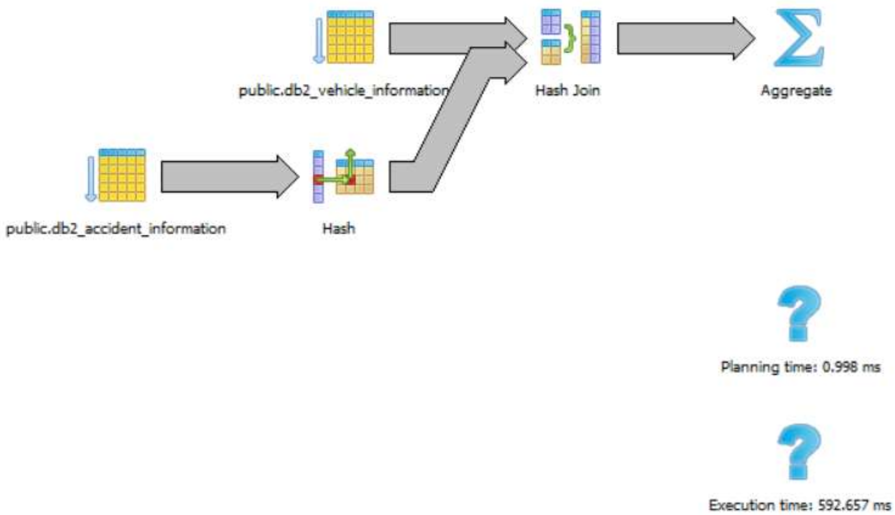
Planning time 0.976ms

Execution time 1176.503ms

time 1.1secs

iv. Βρείτε πόσα ατυχήματα έγιναν σε αγροτική περιοχή το έτος 2012 και η ηλικιακή κατηγορία του οδηγού είναι 36-45.

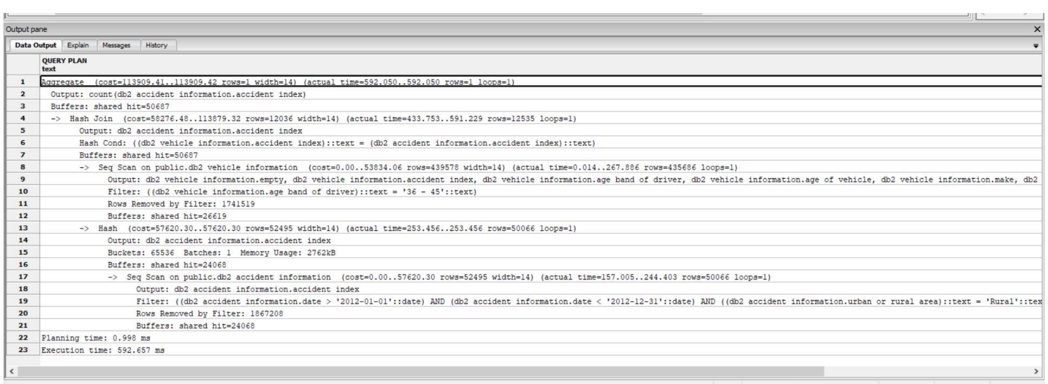




Planning time 0.998ms

Execution time 592.657ms

time 610msec



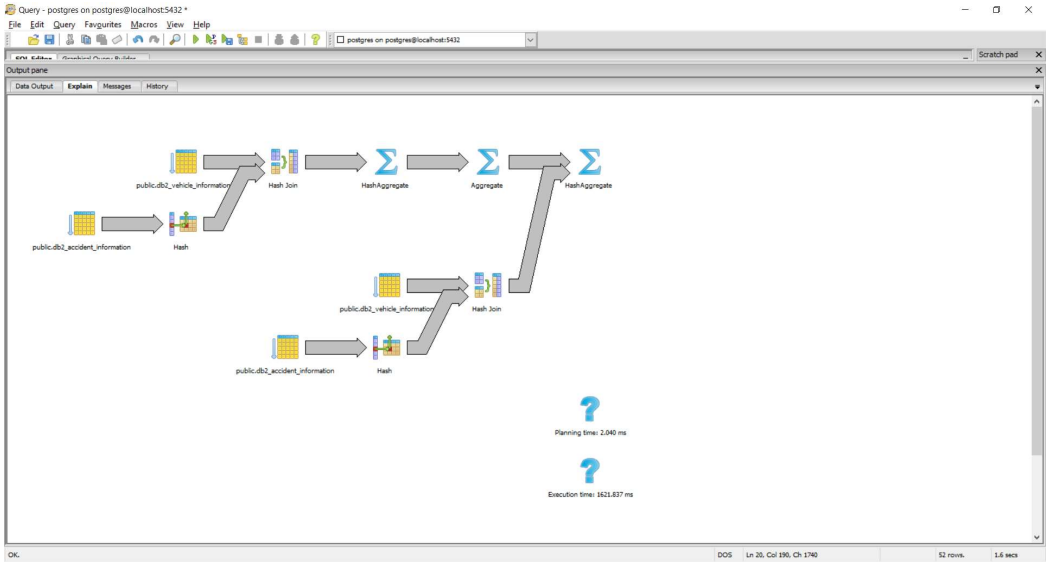
Output pane

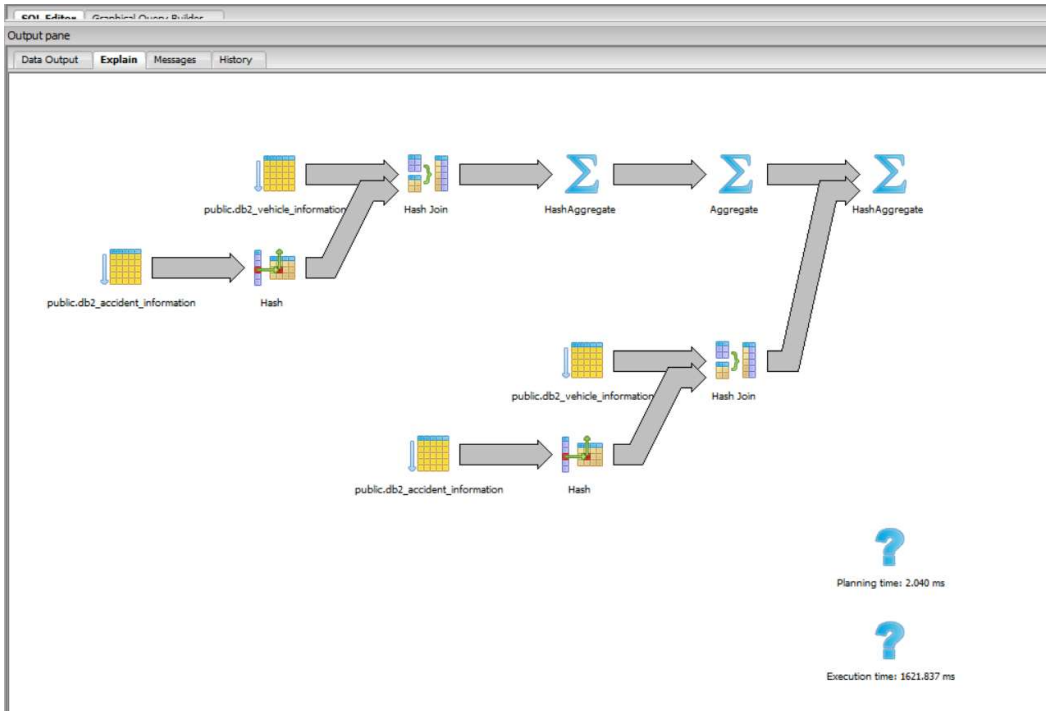
Data Output | Explain | Messages | History

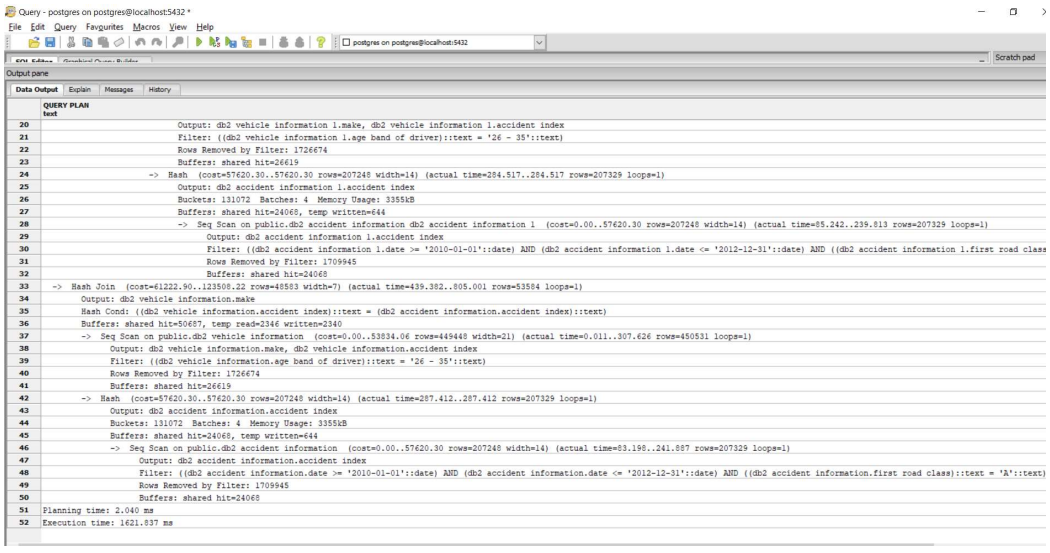
	QUERY PLAN text
1	Aggregate (cost=113909.41..113909.42 rows=1 width=14) (actual time=592.050..592.050 rows=1 loops=1)
2	Output: count(db2 accident information.accident index)
3	Buffers: shared hit=50687
4	-> Hash Join (cost=58276.48..113879.32 rows=12036 width=14) (actual time=433.753..591.229 rows=12535 loops=1)
5	Output: db2 accident information.accident index
6	Hash Cond: ((db2 vehicle information.accident index)::text = (db2 accident information.accident index)::text)
7	Buffers: shared hit=50687
8	-> Seq Scan on public.db2 vehicle information (cost=0.00..53834.06 rows=439578 width=14) (actual time=0.014..267.886 rows=435686 loops=1)
9	Output: db2 vehicle information.empty, db2 vehicle information.accident index, db2 vehicle information.age band of driver, db2 vehicle information.age of vehicle, db2 vehicle information.make, db2
10	Filter: ((db2 vehicle information.age band of driver)::text = '36 - 45'::text)
11	Rows Removed by Filter: 1741519
12	Buffers: shared hit=26619
13	-> Hash (cost=57620.30..57620.30 rows=52495 width=14) (actual time=253.456..253.456 rows=50066 loops=1)
14	Output: db2 accident information.accident index
15	Buckets: 65536 Batches: 1 Memory Usage: 2762kB
16	Buffers: shared hit=24068
17	-> Seq Scan on public.db2 accident information (cost=0.00..57620.30 rows=52495 width=14) (actual time=157.005..244.403 rows=50066 loops=1)
18	Output: db2 accident information.accident index
19	Filter: ((db2 accident information.date > '2012-01-01'::date) AND (db2 accident information.date < '2012-12-31'::date) AND ((db2 accident information.urban or rural area)::text = 'Rural'::tex
20	Rows Removed by Filter: 1867208
21	Buffers: shared hit=24068
22	Planning time: 0.998 ms
23	Execution time: 592.657 ms

OK. DOS Ln 5, Col 209, Ch 456 23 rows. 610 msec

v. Βρείτε τον κατασκευαστή του οχήματος που έχει τα περισσότερα ατυχήματα μεταξύ των ετών 2010 και 2012, η ηλικιακή κατηγορία των οδηγών είναι 26- 35 και η κατηγορία δρόμου είναι Α.







```
Query - postgres on postgres@localhost:5432 *
File  Edit  Query  Favourites  Macros  View  Help
Output pane
Data Output | Explain | Messages | History

QUERY PLAN
text
                        Output: db2 vehicle information 1.make, db2 vehicle information 1.accident index
                        Filter: ((db2 vehicle information 1.age band of driver)::text = '26 - 35'::text)
                        Rows Removed by Filter: 1726674
                        Buffers: shared hit=26619
                  ->  Hash  (cost=57620.30..57620.30 rows=207248 width=14) (actual time=284.517..284.517 rows=207329 loops=1)
                        Output: db2 accident information 1.accident index
                        Buckets: 131072  Batches: 4  Memory Usage: 3355kB
                        Buffers: shared hit=24068, temp written=644
                        ->  Seq Scan on public.db2 accident information db2 accident information 1  (cost=0.00..57620.30 rows=207248 width=14) (actual time=85.242..239.813 rows=207329 loops=1)
                              Output: db2 accident information 1.accident index
                              Filter: ((db2 accident information 1.date >= '2010-01-01'::date) AND (db2 accident information 1.date <= '2012-12-31'::date) AND ((db2 accident information 1.first road class
                              Rows Removed by Filter: 1709945
                              Buffers: shared hit=24068
->  Hash Join  (cost=61222.90..123508.22 rows=48583 width=7) (actual time=439.382..805.001 rows=53584 loops=1)
      Output: db2 vehicle information.make
      Hash Cond: ((db2 vehicle information.accident index)::text = (db2 accident information.accident index)::text)
      Buffers: shared hit=50687, temp read=2346 written=2340
      ->  Seq Scan on public.db2 vehicle information  (cost=0.00..53834.06 rows=449448 width=21) (actual time=0.011..307.626 rows=450531 loops=1)
            Output: db2 vehicle information.make, db2 vehicle information.accident index
            Filter: ((db2 vehicle information.age band of driver)::text = '26 - 35'::text)
            Rows Removed by Filter: 1726674
            Buffers: shared hit=26619
      ->  Hash  (cost=57620.30..57620.30 rows=207248 width=14) (actual time=287.412..287.412 rows=207329 loops=1)
            Output: db2 accident information.accident index
            Buckets: 131072  Batches: 4  Memory Usage: 3355kB
            Buffers: shared hit=24068, temp written=644
            ->  Seq Scan on public.db2 accident information  (cost=0.00..57620.30 rows=207248 width=14) (actual time=83.198..241.887 rows=207329 loops=1)
                  Output: db2 accident information.accident index
                  Filter: ((db2 accident information.date >= '2010-01-01'::date) AND (db2 accident information.date <= '2012-12-31'::date) AND ((db2 accident information.first road class)::text = 'A'::text)
                  Rows Removed by Filter: 1709945
                  Buffers: shared hit=24068
Planning time: 2.040 ms
Execution time: 1621.837 ms
```

Planning time 2.040ms

Execution time 1621.837ms

time 1.6sec

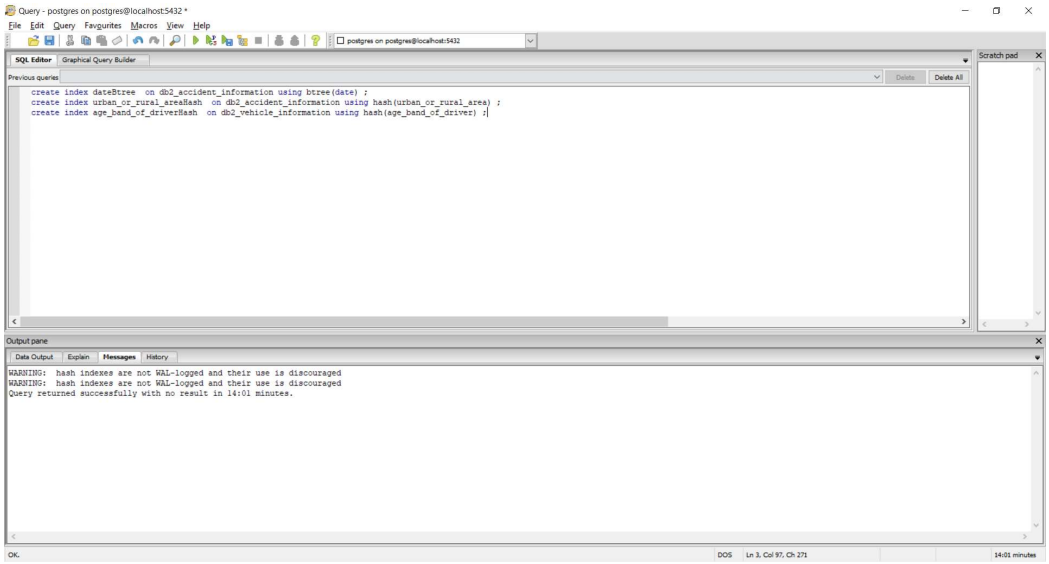
Max parallel workers=16

Query	Planning time	Execution time	Total time
i	0.149ms	469.450ms	485msecs
ii	0.147ms	450.558ms	671msecs
iii	0.976ms	1176.503ms	1.1secs
iv	0.998ms	592.657ms	610msecs
v	2.040ms	1621.837ms	1.6secs

Query	Planning time	Planning time share buffers 16GB	**Planning time max parallel workers 16**	Execution time share buffers 16GB	Execution time share buffers 16GB	**Execution time max parallel workers 16**	Total time	Total time share buffers 16GB	**Total time share buffers 16GB**
i	0.061ms	0.172ms	**0.149ms**	449.906ms	490.710ms	**469.450ms**	469msecs	516msecs	**485msecs**
ii	0.147ms	0.147ms	**0.147ms**	620.265ms	644.286ms	**450.558ms**	641msecs	652msecs	**671msecs**
iii	0.977ms	0.993ms	**0.976ms**	1061.916ms	1147.315ms	**1176.503ms**	1secs	1.1secs	**1.1secs**
iv	0.983ms	1.013ms	**0.998ms**	557.107ms	587.034ms	**592.657ms**	578msecs	592msecs	**610msecs**
v	0.709ms	1.943ms	**2.040ms**	1574.724ms	1648.705ms	**1621.837ms**	1.5secs	1.6secs	**1.6secs**

Τα αποτελέσματα που συγκρίνουμε είναι με έντονα μαύρα γράμματα. Αυτό που μπορούμε να παρατηρήσουμε είναι ότι υπάρχει μια βελτίωση χρόνου σε σχέση με το προηγούμενο παράδειγμα που χρησιμοποιούμε για shared buffers =16GB αλλά οι χρόνοι μας δεν είναι τόσο βελτιωμένη όσο όταν δουλεύαμε όταν είμασταν μέσα στην ram. Αυτό είναι φυσιολογικό γιατί έχουμε αυξήσει τους max_parallel_workers οπότε περιμέναμε μια βελτίωση χρόνου αλλά όχι τόσο μεγάλη όσο όσο θα ήταν αν εργαζόμασταν μέσα στη ram.

(d) Δημιουργήστε τα κατάλληλα ευρετήρια στη ΒΔ για να τρέξουν οι παραπάνω επερωτήσεις πιο γρήγορα. Για κάθε ευρετήριο που θα δημιουργήσετε θα εξηγήσετε τους λόγους για τους οποίους επιλέξατε τον συγκεκριμένο τύπο ευρετηρίου και το πώς βοηθάει στην βελτίωση του χρόνου εκτέλεσης.



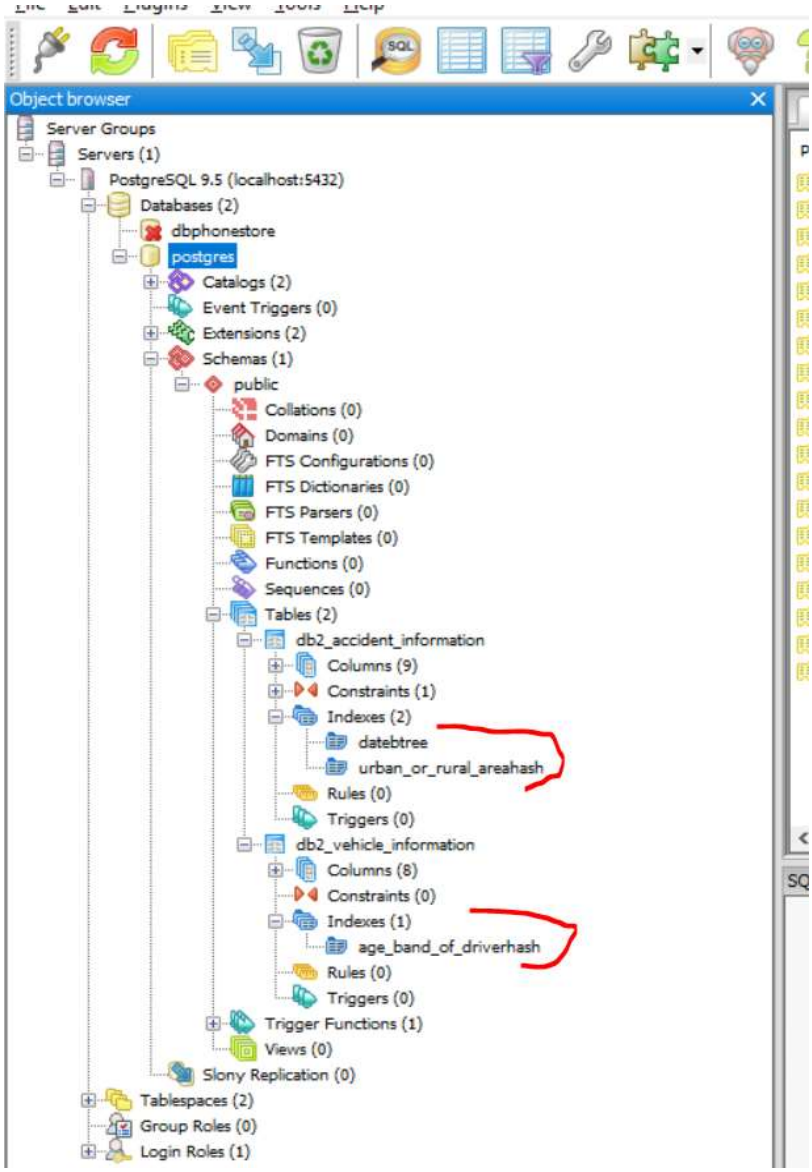
Θα δημιουργήσω ένα btree ευρετήριο για την στήλη **date** απο τον πίνακα **db2_accident_information.**Χρησιμοποιώ btrees γιατί διαχειρίζονται καλύτερα ισότητες και οτιδήποτε έχει να κάνει με εύρος. Η postgresql προτιμάει να χρησιμοποιήσει btrees όταν μια στήλη εμπλέκεται σε συγκρίσεις. Δηλαδή όταν βλέπουμε τα παρακάτω σύμβολα.

μικροτερο(<=) ,μικροτερο ισο (<),μεγαλυτερο(>) ,μεγαλυτερο ισο(>=)και ισοτητες.

create index dateBtree on db2_accident_information using btree(date) ;

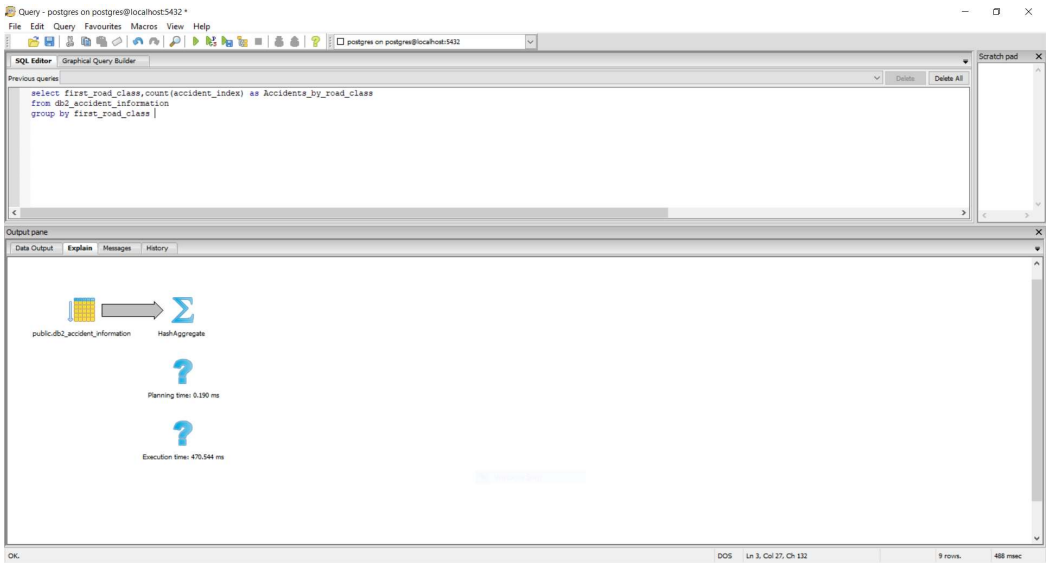
Στην συνέχεια δημιουργώ hash ευρετήρια για την στήλη **urban_or_rural_area** του πίνακα **db2_accident_information** και για την στήλη **age_band_of_driver** του πίνακα **db2_vehicle_information.**

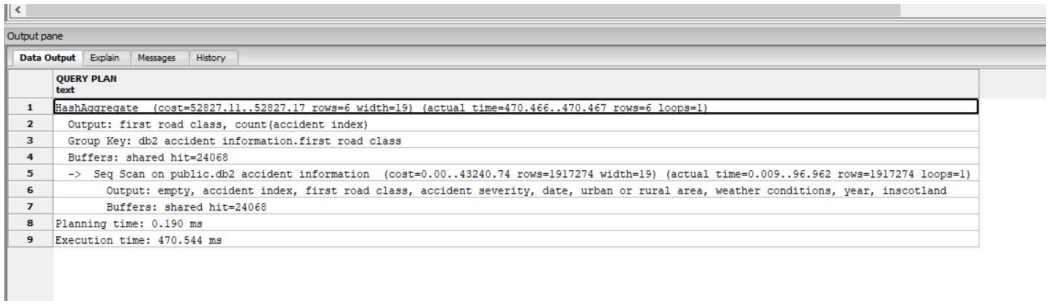
Ό χρόνος που μας πήρε να φτιάξουμε τα ευρετήρια είναι 14 λεπτά και 01 δευτερόλεπτα. Μας πήρε τοσο χρόνο γιατί τα hash ευρετήρια χρειάζονται αρκετό χρόνο για να δημιουργηθούν.


Object browser
Server Groups
Servers (1)
PostgreSQL 9.5 (localhost:5432)
Databases (2)
dbphonestore
postgres
Catalogs (2)
Event Triggers (0)
Extensions (2)
Schemas (1)
public
Collations (0)
Domains (0)
FTS Configurations (0)
FTS Dictionaries (0)
FTS Parsers (0)
FTS Templates (0)
Functions (0)
Sequences (0)
Tables (2)
db2_accident_information
Columns (9)
Constraints (1)
Indexes (2)
datebtree
urban_or_rural_areahash
Rules (0)
Triggers (0)
db2_vehicle_information
Columns (8)
Constraints (0)
Indexes (1)
age_band_of_driverhash
Rules (0)
Triggers (0)
Trigger Functions (1)
Views (0)
Slony Replication (0)
Tablespaces (2)
Group Roles (0)
Login Roles (1)

Τρέχω ξανά τα επερωτήματα.

i. Βρείτε πόσα ατυχήματα έγιναν ανά κατηγορία δρόμου (road class).



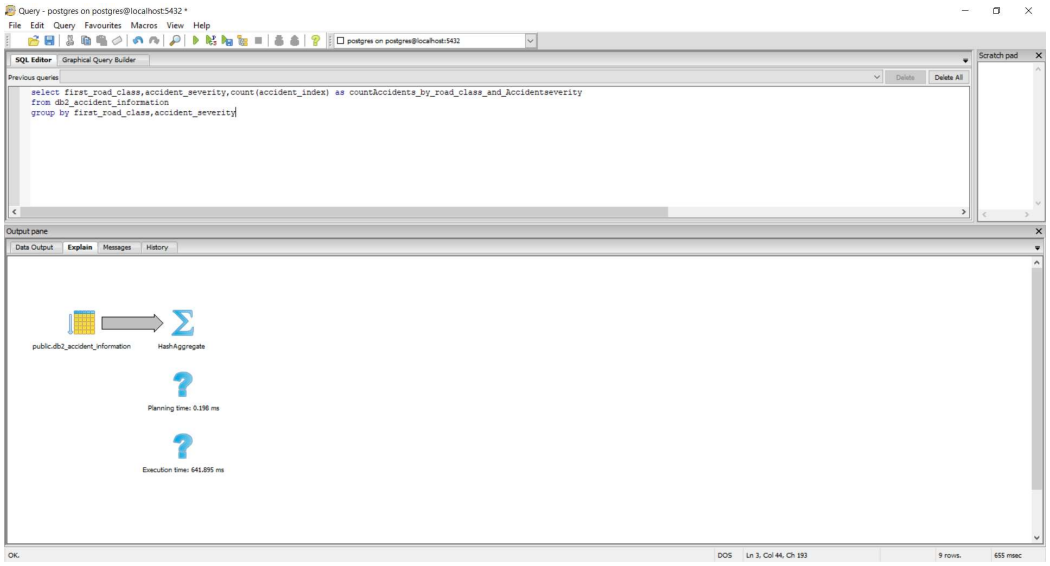


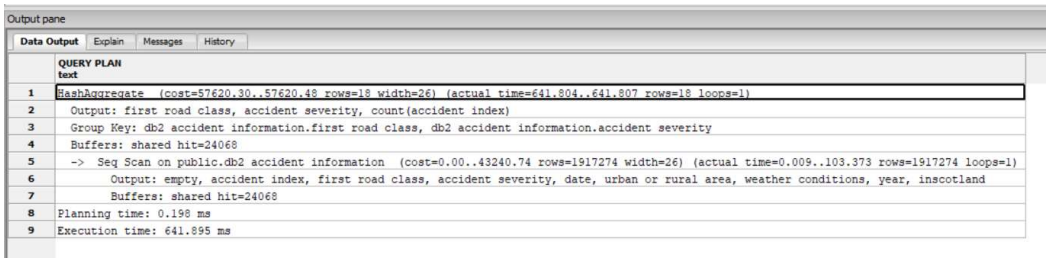
Planning time:0.190ms

Execution time:470.544ms

Time:488msec

ii. Πόσα είναι τα ατυχήματα ανά κατηγορία δρόμου και ανά κατηγορία ατυχήματος (Accident Severity).




Output pane

Data Output | Explain | Messages | History

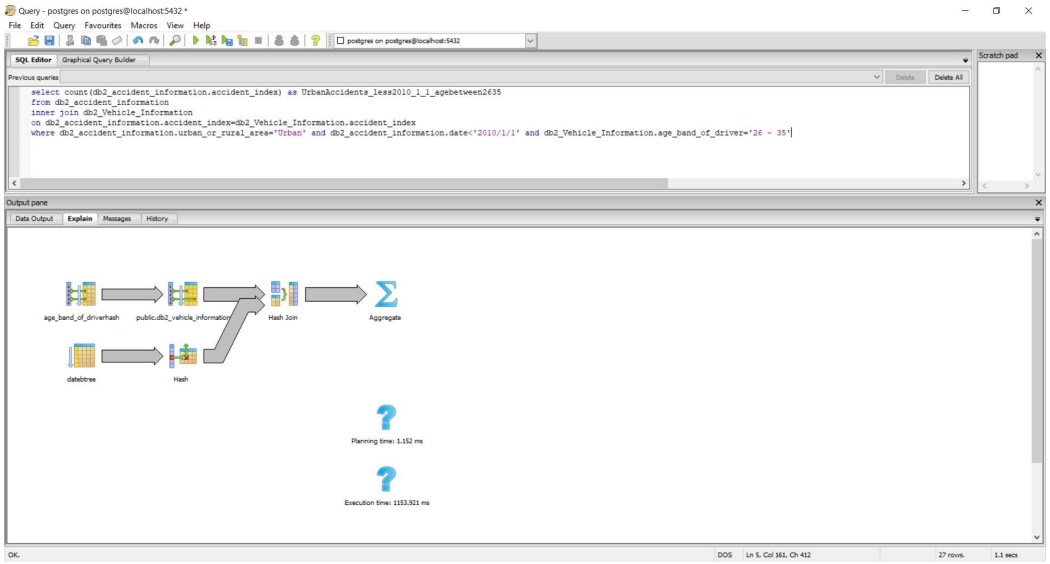
	QUERY PLAN text
1	HashAggregate (cost=57620.30..57620.48 rows=18 width=26) (actual time=641.804..641.807 rows=18 loops=1)
2	Output: first road class, accident severity, count(accident index)
3	Group Key: db2 accident information.first road class, db2 accident information.accident severity
4	Buffers: shared hit=24068
5	-> Seq Scan on public.db2 accident information (cost=0.00..43240.74 rows=1917274 width=26) (actual time=0.009..103.373 rows=1917274 loops=1)
6	Output: empty, accident index, first road class, accident severity, date, urban or rural area, weather conditions, year, inscotland
7	Buffers: shared hit=24068
8	Planning time: 0.198 ms
9	Execution time: 641.895 ms

Planning time:0.198ms

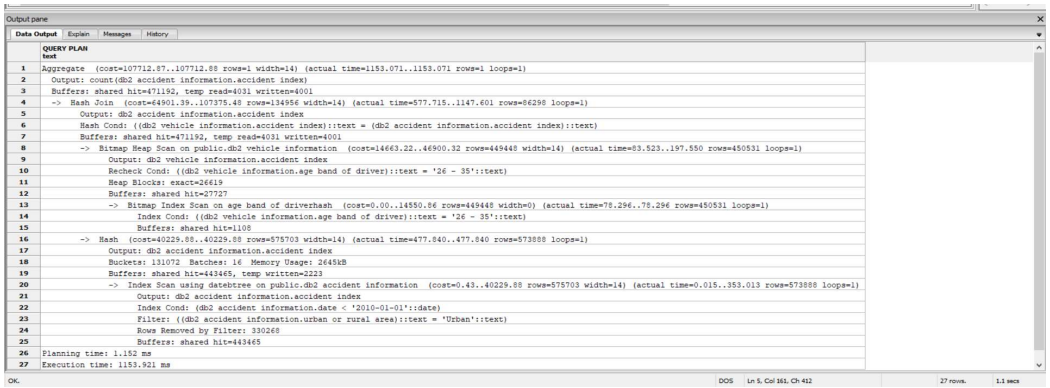
Execution time:641.895ms

time:655msec

iii. Βρείτε πόσα ατυχήματα έγιναν σε αστική περιοχή πριν από την 1/1/2010 και η ηλικιακή κατηγορία του οδηγού είναι 26-35.



Όπως βλέπουμε έχει χρησιμοποιήσει τα ευρετήρια μας **urban_or_rural_areaHash** και **dateBtree**

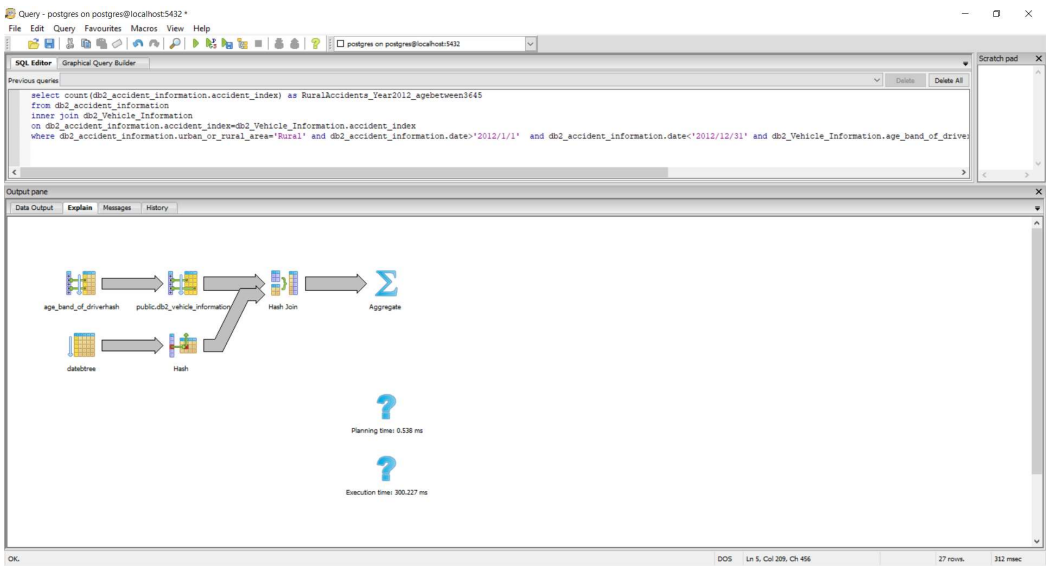


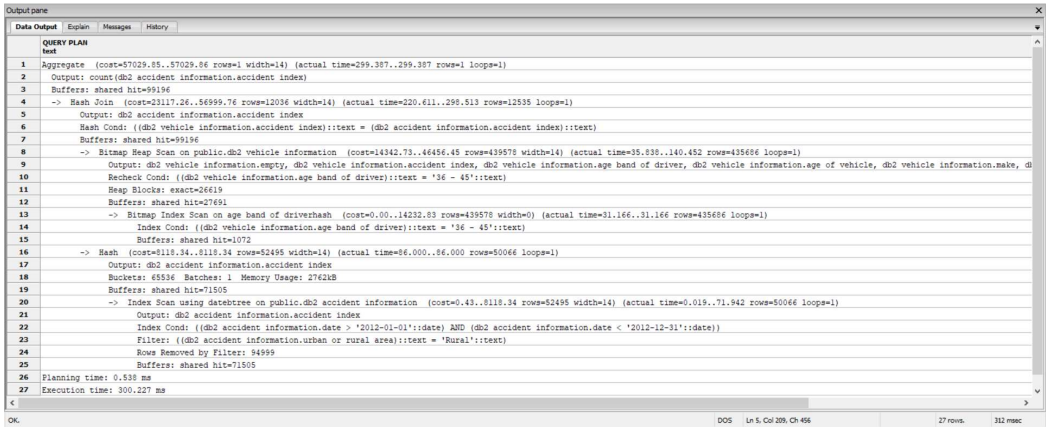
Planning time:1.152ms

Execution time:1153.921ms

time:1.1secs

iv. Βρείτε πόσα ατυχήματα έγιναν σε αγροτική περιοχή το έτος 2012 και η ηλικιακή κατηγορία του οδηγού είναι 36-45.





Output pane

Data Output | Explain | Messages | History

	QUERY PLAN text
1	Aggregate (cost=57029.85..57029.86 rows=1 width=14) (actual time=299.387..299.387 rows=1 loops=1)
2	Output: count(db2 accident information.accident index)
3	Buffers: shared hit=99196
4	-> Hash Join (cost=23117.26..56999.76 rows=12036 width=14) (actual time=220.611..298.513 rows=12535 loops=1)
5	Output: db2 accident information.accident index
6	Hash Cond: ((db2 vehicle information.accident index)::text = (db2 accident information.accident index)::text)
7	Buffers: shared hit=99196
8	-> Bitmap Heap Scan on public.db2 vehicle information (cost=14342.73..46456.45 rows=439578 width=14) (actual time=35.838..140.452 rows=435686 loops=1)
9	Output: db2 vehicle information.empty, db2 vehicle information.accident index, db2 vehicle information.age band of driver, db2 vehicle information.age of vehicle, db2 vehicle information.make, d
10	Recheck Cond: ((db2 vehicle information.age band of driver)::text = '36 - 45'::text)
11	Heap Blocks: exact=26619
12	Buffers: shared hit=27691
13	-> Bitmap Index Scan on age band of driverhash (cost=0.00..14232.83 rows=439578 width=0) (actual time=31.166..31.166 rows=435686 loops=1)
14	Index Cond: ((db2 vehicle information.age band of driver)::text = '36 - 45'::text)
15	Buffers: shared hit=1072
16	-> Hash (cost=8118.34..8118.34 rows=52495 width=14) (actual time=86.000..86.000 rows=50066 loops=1)
17	Output: db2 accident information.accident index
18	Buckets: 65536 Batches: 1 Memory Usage: 2762kB
19	Buffers: shared hit=71505
20	-> Index Scan using datebtree on public.db2 accident information (cost=0.43..8118.34 rows=52495 width=14) (actual time=0.019..71.942 rows=50066 loops=1)
21	Output: db2 accident information.accident index
22	Index Cond: ((db2 accident information.date > '2012-01-01'::date) AND (db2 accident information.date < '2012-12-31'::date))
23	Filter: ((db2 accident information.urban or rural area)::text = 'Rural'::text)
24	Rows Removed by Filter: 94999
25	Buffers: shared hit=71505
26	Planning time: 0.538 ms
27	Execution time: 300.227 ms

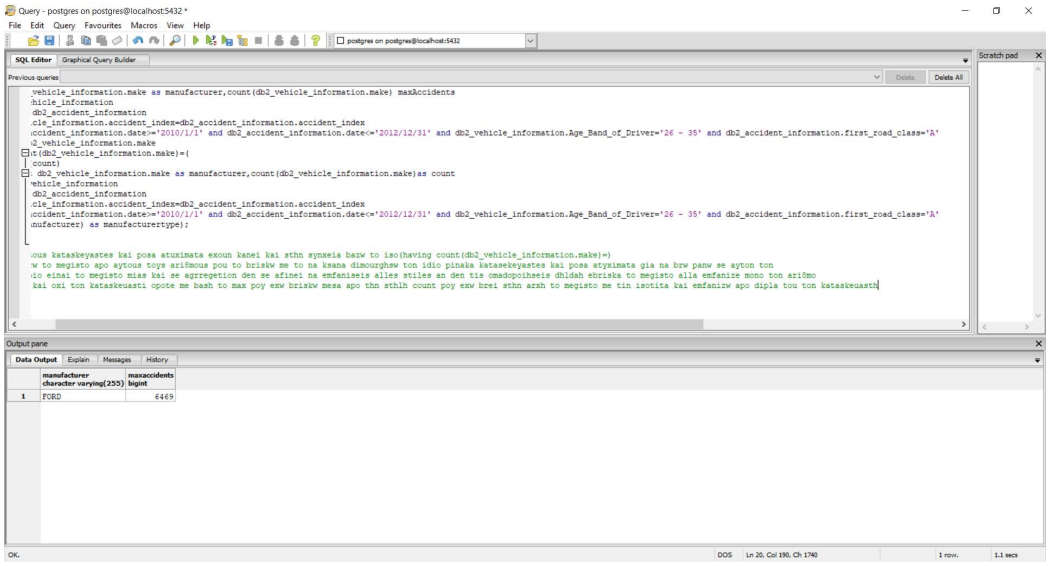
OK. DOS Ln 5, Col 209, Ch 456 27 rows. 312 msec

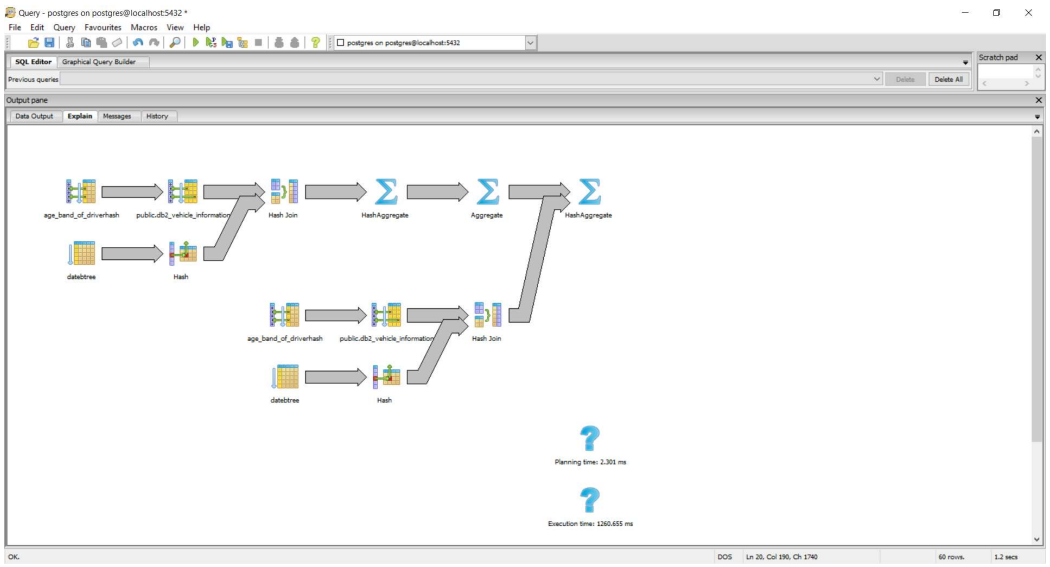
Planning time 0.538ms

Execution time:300.227ms

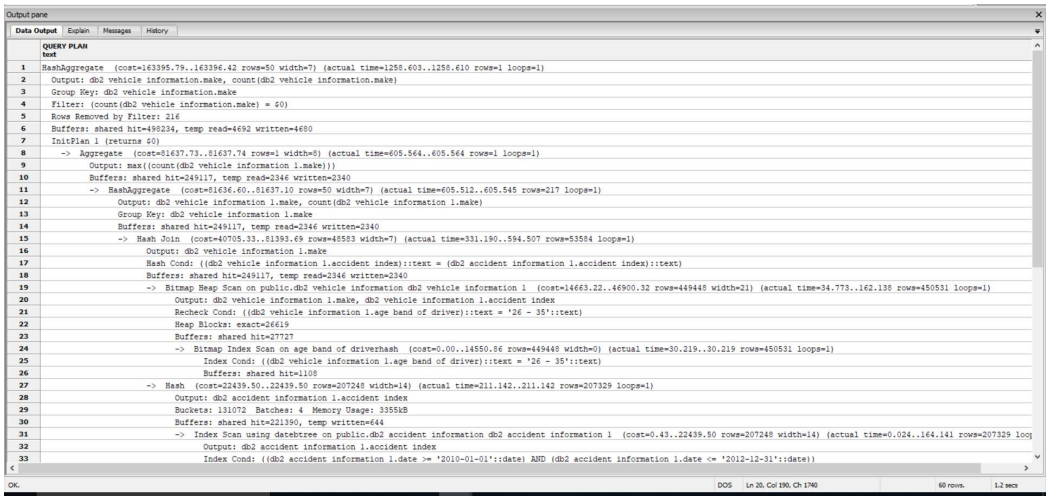
time:312msec

v. Βρείτε τον κατασκευαστή του οχήματος που έχει τα περισσότερα ατυχήματα μεταξύ των ετών 2010 και 2012, η ηλικιακή κατηγορία των οδηγών είναι 26- 35 και η κατηγορία δρόμου είναι Α.

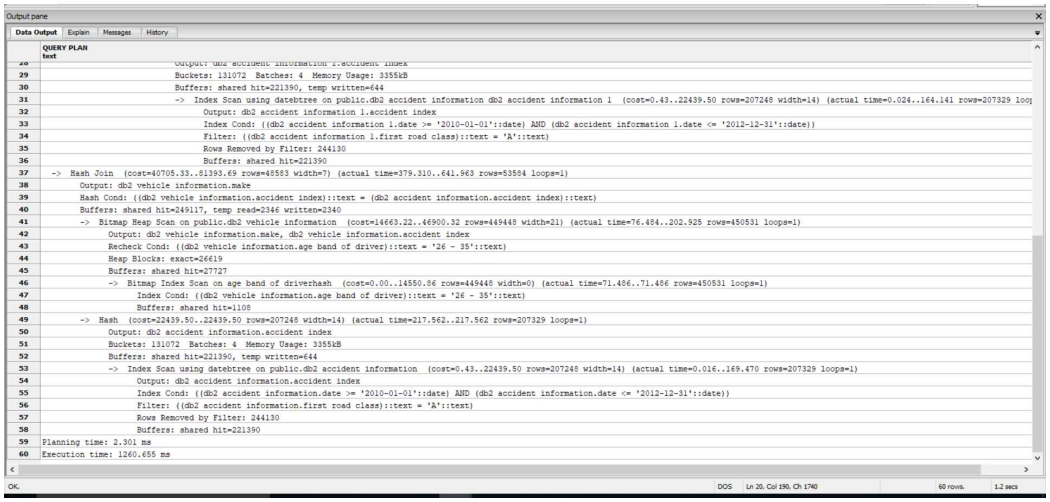




Όπως βλέπουμε χρησιμοποιεί στην επεξεργασία το **datebtree** και **age_band_of_driverHash**



```
QUERY PLAN
text
HashAggregate  (cost=163395.79..163396.42 rows=50 width=7) (actual time=1258.603..1258.610 rows=1 loops=1)
  Output: db2 vehicle information.make, count(db2 vehicle information.make)
  Group Key: db2 vehicle information.make
  Filter: (count(db2 vehicle information.make) = 60)
  Rows Removed by Filter: 216
  Buffers: shared hit=498234, temp read=4692 written=4680
  InitPlan 1 (returns $0)
    ->  Aggregate  (cost=81637.73..81637.74 rows=1 width=8) (actual time=605.564..605.564 rows=1 loops=1)
          Output: max((count(db2 vehicle information 1.make)))
          Buffers: shared hit=249117, temp read=2346 written=2340
          ->  HashAggregate  (cost=81636.60..81637.10 rows=50 width=7) (actual time=605.512..605.545 rows=217 loops=1)
                Output: db2 vehicle information 1.make, count(db2 vehicle information 1.make)
                Group Key: db2 vehicle information 1.make
                Buffers: shared hit=249117, temp read=2346 written=2340
                ->  Hash Join  (cost=40705.33..81393.69 rows=48583 width=7) (actual time=331.190..594.507 rows=53584 loops=1)
                      Output: db2 vehicle information 1.make
                      Hash Cond: ((db2 vehicle information 1.accident index)::text = (db2 accident information 1.accident index)::text)
                      Buffers: shared hit=249117, temp read=2346 written=2340
                      ->  Bitmap Heap Scan on public.db2 vehicle information db2 vehicle information 1  (cost=14663.22..46900.32 rows=449448 width=21) (actual time=34.773..162.138 rows=450531 loops=1)
                            Output: db2 vehicle information 1.make, db2 vehicle information 1.accident index
                            Recheck Cond: ((db2 vehicle information 1.age band of driver)::text = '26 - 35'::text)
                            Heap Blocks: exact=26619
                            Buffers: shared hit=27727
                            ->  Bitmap Index Scan on age band of driverhash  (cost=0.00..14550.86 rows=449448 width=0) (actual time=30.219..30.219 rows=450531 loops=1)
                                  Index Cond: ((db2 vehicle information 1.age band of driver)::text = '26 - 35'::text)
                                  Buffers: shared hit=1108
                      ->  Hash  (cost=22439.50..22439.50 rows=207248 width=14) (actual time=211.142..211.142 rows=207329 loops=1)
                            Output: db2 accident information 1.accident index
                            Buckets: 131072  Batches: 4  Memory Usage: 3355kB
                            Buffers: shared hit=221390, temp written=644
                            ->  Index Scan using datebtree on public.db2 accident information db2 accident information 1  (cost=0.43..22439.50 rows=207248 width=14) (actual time=0.024..164.141 rows=207329 loo
                                  Output: db2 accident information 1.accident index
                                  Index Cond: ((db2 accident information 1.date >= '2010-01-01'::date) AND (db2 accident information 1.date <= '2012-12-31'::date))
```



```
QUERY PLAN
text
                                  Buckets: 131072  Batches: 4  Memory Usage: 3355kB
                                  Buffers: shared hit=221390, temp written=644
                                  ->  Index Scan using datebtree on public.db2 accident information db2 accident information 1  (cost=0.43..22439.50 rows=207248 width=14) (actual time=0.024..164.141 rows=207329 loo
                                        Output: db2 accident information 1.accident index
                                        Index Cond: ((db2 accident information 1.date >= '2010-01-01'::date) AND (db2 accident information 1.date <= '2012-12-31'::date))
                                        Filter: ((db2 accident information 1.first road class)::text = 'A'::text)
                                        Rows Removed by Filter: 244130
                                        Buffers: shared hit=221390
  ->  Hash Join  (cost=40705.33..81393.69 rows=48583 width=7) (actual time=379.310..641.963 rows=53584 loops=1)
        Output: db2 vehicle information.make
        Hash Cond: ((db2 vehicle information.accident index)::text = (db2 accident information.accident index)::text)
        Buffers: shared hit=249117, temp read=2346 written=2340
        ->  Bitmap Heap Scan on public.db2 vehicle information  (cost=14663.22..46900.32 rows=449448 width=21) (actual time=76.484..202.925 rows=450531 loops=1)
              Output: db2 vehicle information.make, db2 vehicle information.accident index
              Recheck Cond: ((db2 vehicle information.age band of driver)::text = '26 - 35'::text)
              Heap Blocks: exact=26619
              Buffers: shared hit=27727
              ->  Bitmap Index Scan on age band of driverhash  (cost=0.00..14550.86 rows=449448 width=0) (actual time=71.486..71.486 rows=450531 loops=1)
                    Index Cond: ((db2 vehicle information.age band of driver)::text = '26 - 35'::text)
                    Buffers: shared hit=1108
        ->  Hash  (cost=22439.50..22439.50 rows=207248 width=14) (actual time=217.562..217.562 rows=207329 loops=1)
              Output: db2 accident information.accident index
              Buckets: 131072  Batches: 4  Memory Usage: 3355kB
              Buffers: shared hit=221390, temp written=644
              ->  Index Scan using datebtree on public.db2 accident information  (cost=0.43..22439.50 rows=207248 width=14) (actual time=0.016..169.470 rows=207329 loops=1)
                    Output: db2 accident information.accident index
                    Index Cond: ((db2 accident information.date >= '2010-01-01'::date) AND (db2 accident information.date <= '2012-12-31'::date))
                    Filter: ((db2 accident information.first road class)::text = 'A'::text)
                    Rows Removed by Filter: 244130
                    Buffers: shared hit=221390
Planning time: 2.301 ms
Execution time: 1260.655 ms
```

Planning time:2.301ms

Execution time:1260.655ms

Time:1.2secs

index times

Query	Planning time	Execution time	Total time
i	0.190ms	470.544ms	488msec
ii	0.198ms	641.895ms	655msec
iii	1.152ms	1153.921ms	1.1sec
iv	0.538ms	300.277ms	312msec
v	2.031ms	1260.655ms	1.2sec

Query	Planning time	Planning time share buffers 16GB	Planning time max parallel workers 16	**Planning time index times**	Execution time share buffers 16GB	Execution time share buffers 16GB	Execution time max parallel workers 16	**Execution time index times**	Total time	Total time share buffers 16GB	Total time share buffers 16GB	Total time **index times**
i	0.061 ms	0.172 ms	0.149 ms	0.190ms	449.906ms	490.710ms	469.450ms	470.544ms	469msecs	516msecs	485msecs	488msec
ii	0.147 ms	0.147 ms	0.147 ms	0.198ms	620.265ms	644.286ms	450.558ms	641.895ms	641msecs	652msecs	671msecs	655msec
iii	0.977 ms	0.993 ms	0.976 ms	**1.152ms**	1061.916 ms	1147.315 ms	1176.503 ms	**1153.921 ms**	1secs	1.1secs	1.1secs	**1.1sec**
iv	0.983 ms	1.013 ms	0.998 ms	**0.538ms**	557.107ms	587.034ms	592.657ms	**300.277ms**	578msecs	592msecs	610msecs	**312msec**
v	0.709 ms	1.943 ms	2.040 ms	**2.031ms**	1574.724 ms	1648.705 ms	1621.837 ms	**1260.655 ms**	1.5secs	1.6secs	1.6secs	**1.2sec**

Οί μετρήσεις που μας ενδιαφέρουν είναι απο το iii,iv,v ερώτημα όπου χρησιμοποιούμε τα ευρετήρια μας και βλέπουμε οτι έχουμε βελτίωση χρόνου.

Για το τρίτο ερώτημα iii) έχουμε μα μικρή βελτίωση χρόνου απο όταν χρησιμοποιούμε 16 parallel workers.Βέβαια δεν έχουμε την ίδια βελτίωση χρόνου σε σχέση με όταν ήμασταν μέσα στην ram και όταν έχουμε shared buffers 16GB.

Για το τέταρτο ερώτημα iv) έχουμε βελτίωση χρόνου απο όλα τα υπόλοιπα ερωτήματα ακόμα και απο όταν βρισκόμαστε μέσα στην ram.Έχουμε χρόνο **312msec 578 μέσα στην ram,έχουμε ρίξει τον χρόνο στο ήμισυ.**

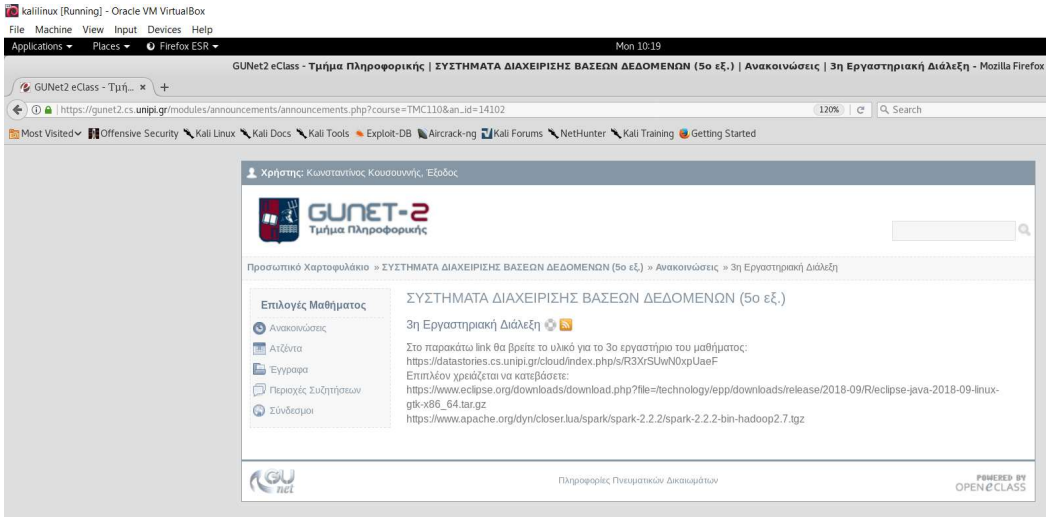
Για το πέμπτο ερώτημα v) έχουμε βελτίωση χρόνου απο όλα τα υπόλοιπα ερωτήματα ακόμα και απο όταν βρισκόμαστε μέσα στην ram. Έχουμε χρόνο **1.2sec ένω 1.5msec μέσα στην ram,έχουμε ρίξει το χρόνο 0.3sec.**

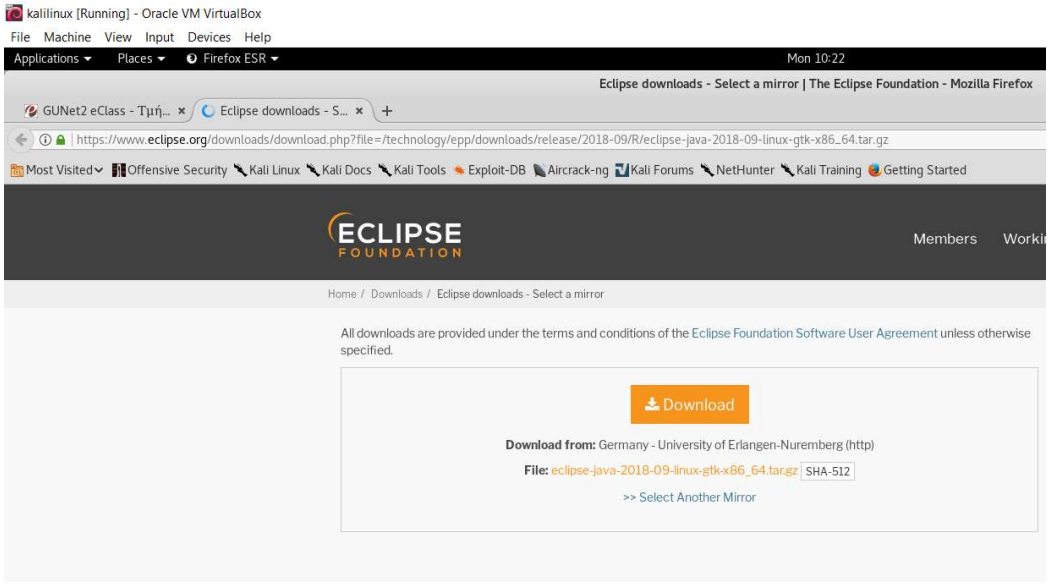
Ερώτημα 2 (40%). Υλοποίηση Ερωτημάτων σε Spark (RDD)

Αφού φορτώσετε το σύνολο δεδομένων που σας δίνεται στο Spark:

Για να φωρτόσουμε τα δεδομένα αρχικά κάνουμε εγκατάσταση το eclipse.Το eclipse θα το κάνουμε εγκατάσταση σε kali linux.Έχω κάνει εγκατάσταση το μηχάνημα σε virtual box.

Κατεβάζω απο την ιστοσελίδα του μαθήματος το αντίστοιχο virtual box και το hadoop για τις βιβλιοθήκες του spark.



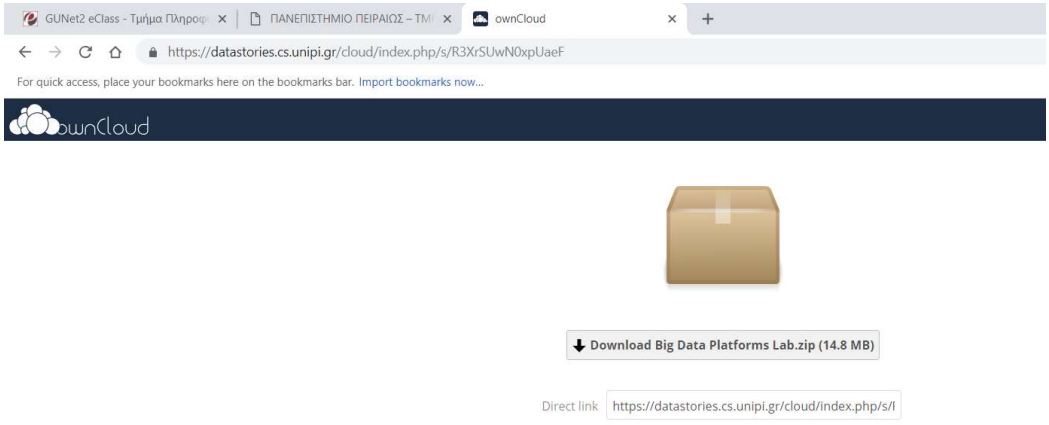


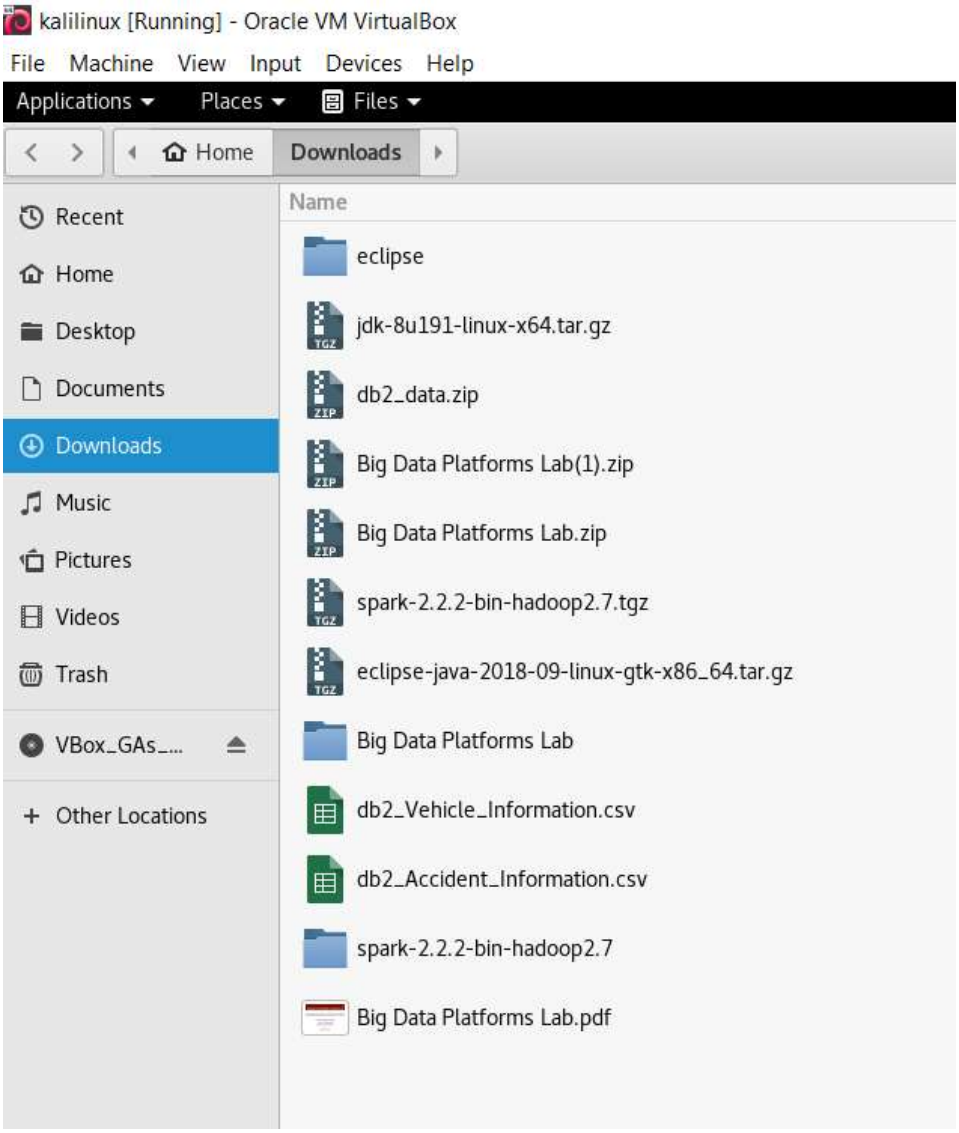
Κατεβάζουμε το αρχείο και κάνουμε εγκατάσταση το eclipse.

Στην συνέχεια κατεβάζουμε τις βιβλιοθήκες μας για το spark.

Επίσης έχουμε κάνει εγκατάσταση και την java έκδοση 1.8

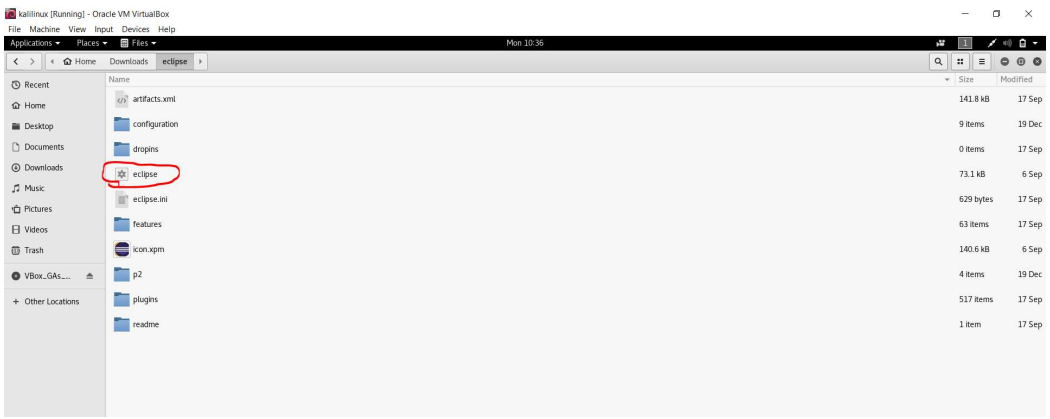
Τέλος κατεβάζουμε και τον κώδικα που μας έχει δοθεί από το εργαστήριο.

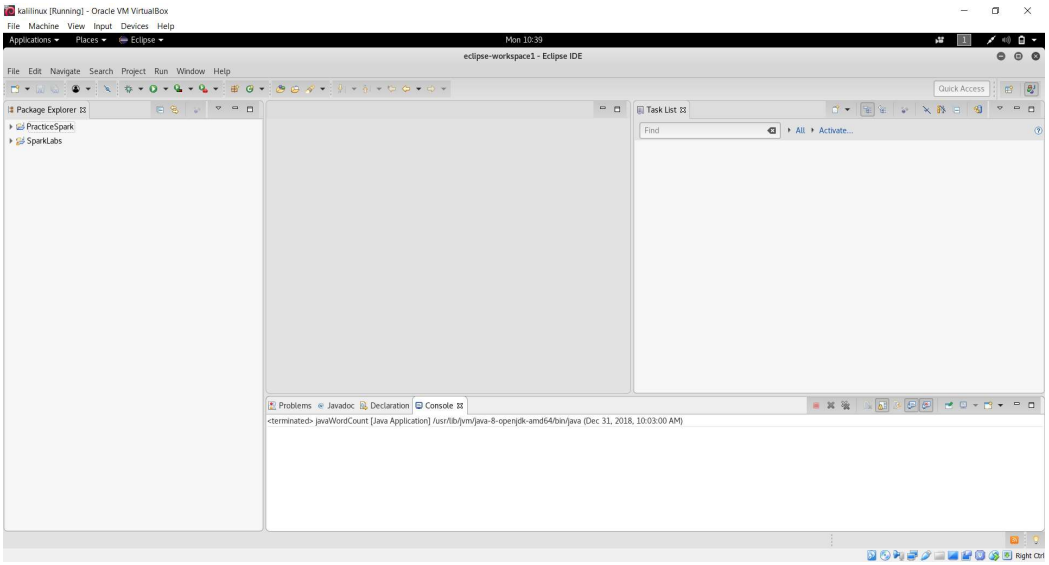



kalilinux [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Files
Home Downloads
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
VBox_GAs_...
Other Locations
Name
eclipse
jdk-8u191-linux-x64.tar.gz
db2_data.zip
Big Data Platforms Lab(1).zip
Big Data Platforms Lab.zip
spark-2.2.2-bin-hadoop2.7.tgz
eclipse-java-2018-09-linux-gtk-x86_64.tar.gz
Big Data Platforms Lab
db2_Vehicle_Information.csv
db2_Accident_Information.csv
spark-2.2.2-bin-hadoop2.7
Big Data Platforms Lab.pdf

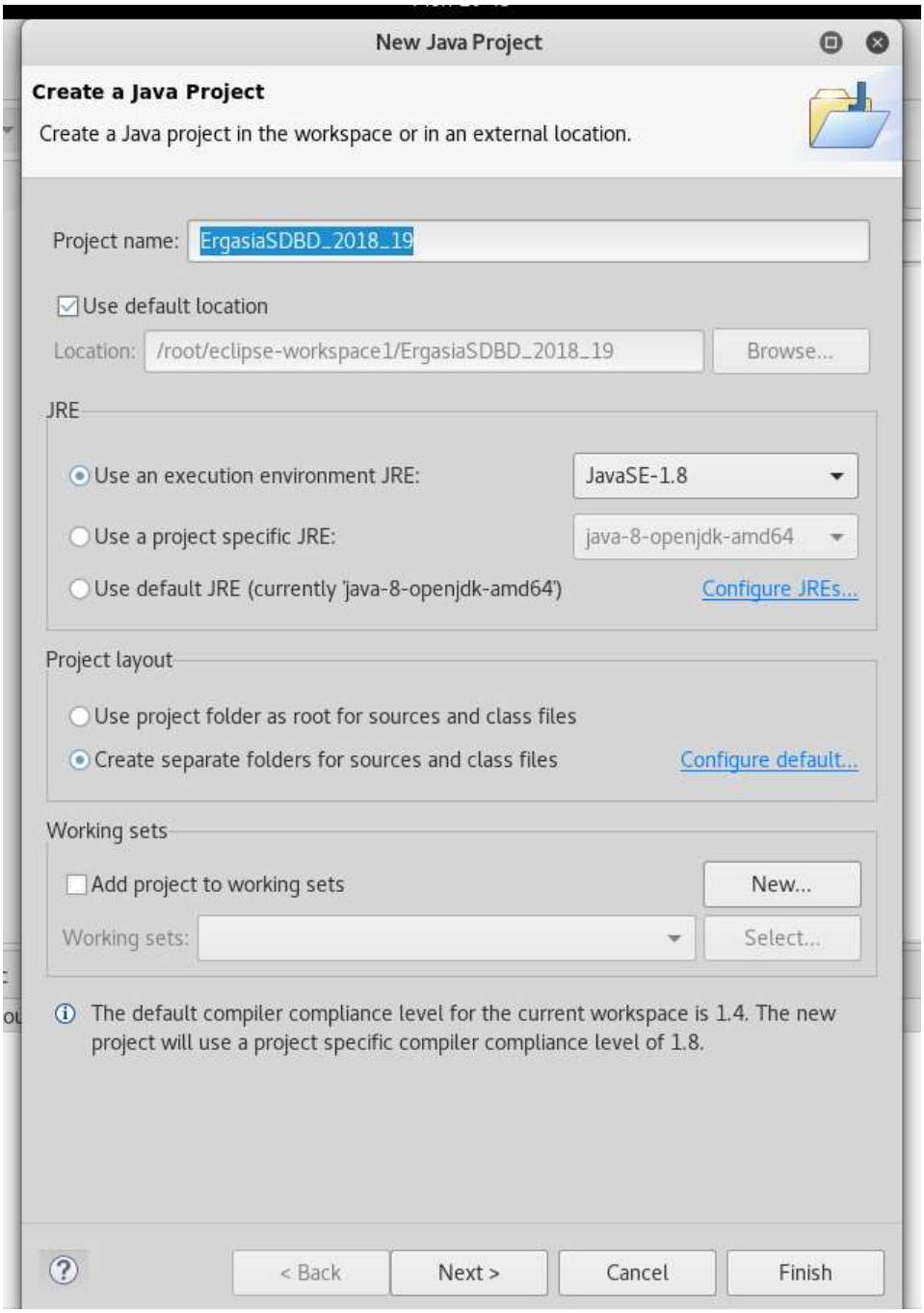
Και τέλος κατεβάζουμε και τα δυο αρχεία με τα οποία δουλεύουμε.

Για να ανοίξουμε το eclipse ανοίγουμε στον αντίστοιχο φάκελο και πατάμε αριστερό κλικ στο eclipse.exe

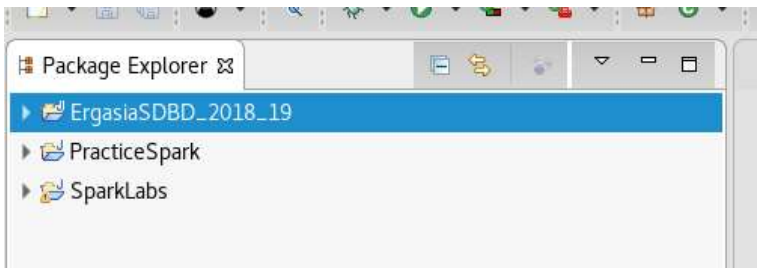




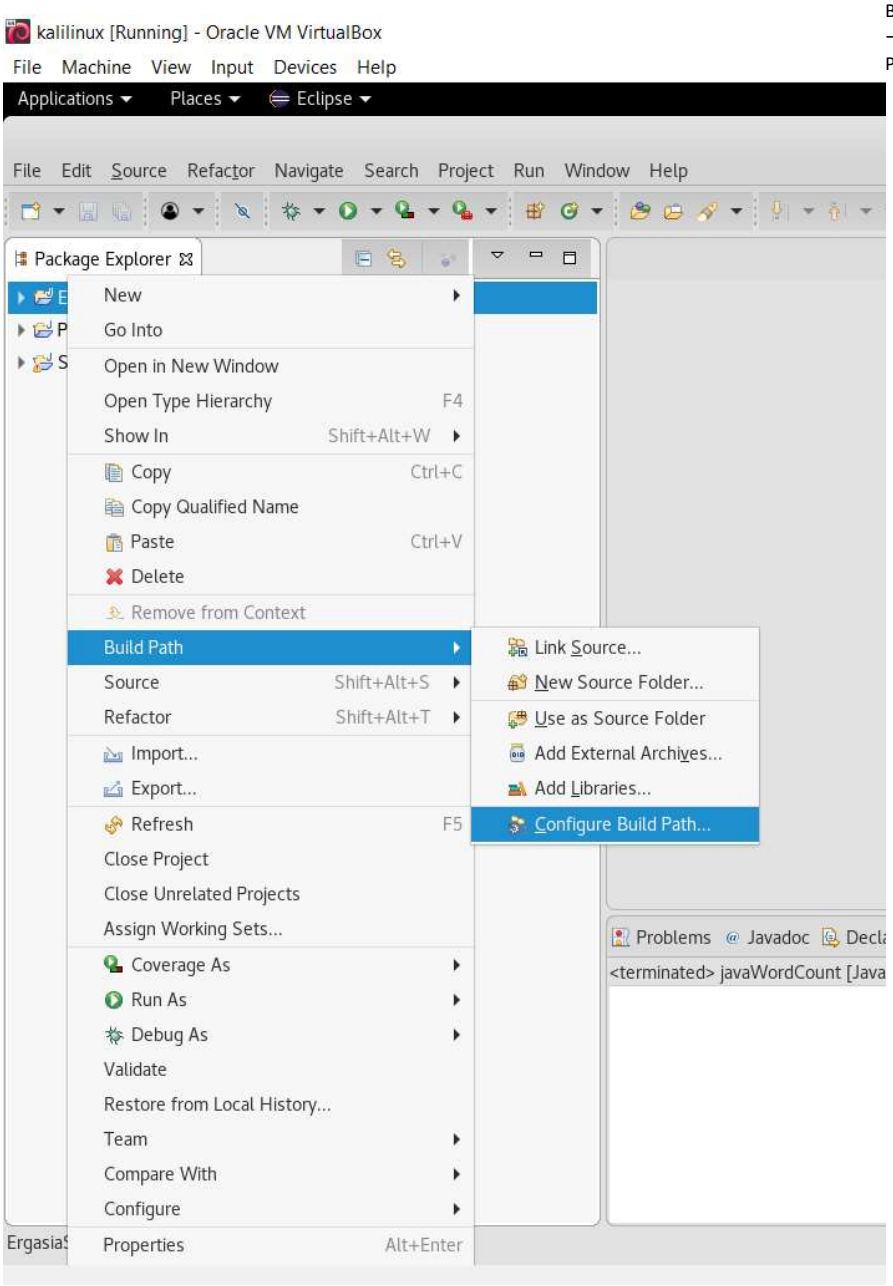
Δημιουργώ ένα νέο Project με όνομα **ErgasiaSDB_2018_19**


New Java Project
Create a Java Project
Create a Java project in the workspace or in an external location.
Project name: ErgasiaSDBD_2018_19
Use default location
Location: /root/eclipse-workspace1/ErgasiaSDBD_2018_19
Browse...
JRE
Use an execution environment JRE:
JavaSE-1.8
Use a project specific JRE:
java-8-openjdk-amd64
Use default JRE (currently 'java-8-openjdk-amd64')
Configure JREs...
Project layout
Use project folder as root for sources and class files
Create separate folders for sources and class files
Configure default...
Working sets
Add project to working sets
New...
Working sets:
Select...
The default compiler compliance level for the current workspace is 1.4. The new project will use a project specific compiler compliance level of 1.8.
< Back
Next >
Cancel
Finish

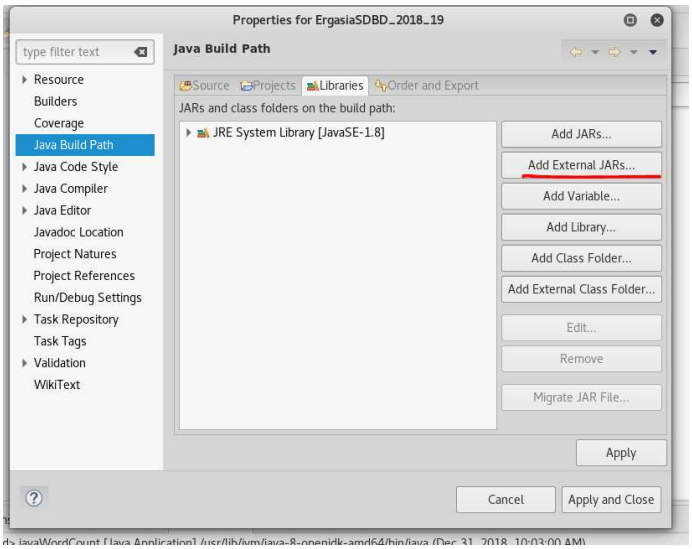
Το project μας έχει δημιουργηθεί



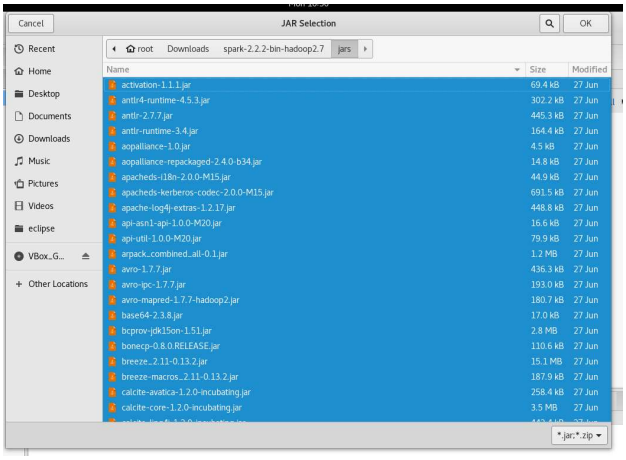
Κανουμε δεξι κλικ Build Path →Configure Build Path



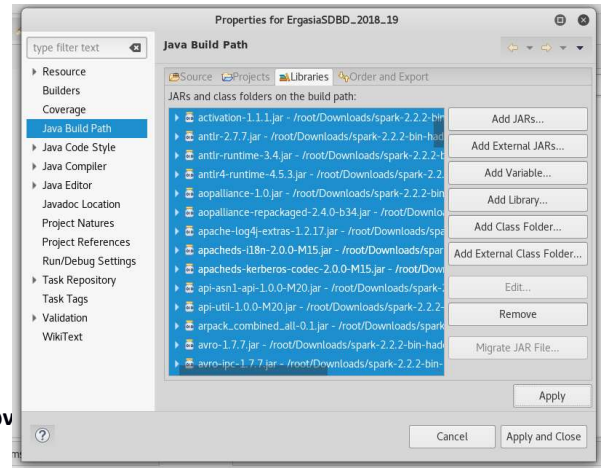
Add External jars



Επιλέγουμε όλες τις βιβλιοθήκες μας και πατάμε **ok** από το αρχείο που κατεβάσαμε προηγούμενος για να κατεβάσουμε τις βιβλιοθήκες από το spark.



Τέλος πατάμε **aplly** για να κάνουμε save στις αλλαγές μας.



Συστήματα Διαχείρισης Βάσεων Δεδομένων

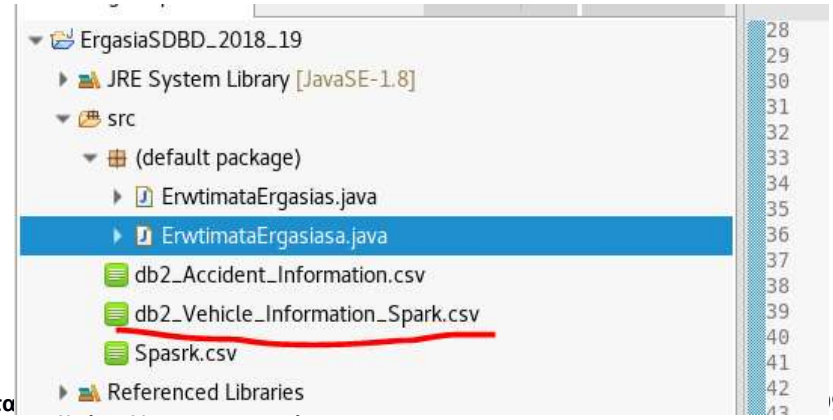
(a) Υπολογίστε κάποια στατιστικά για τα δεδομένα (ενδεικτικά, μπορείτε να υπολογίσετε πόσοι είναι οι κατασκευαστές των αυτοκινήτων, πόσα ατυχήματα έγιναν ανά έτος, κλπ.).

Θα υπολογίσω πόσοι είναι οι κατασκευαστές των αυτοκινήτων.

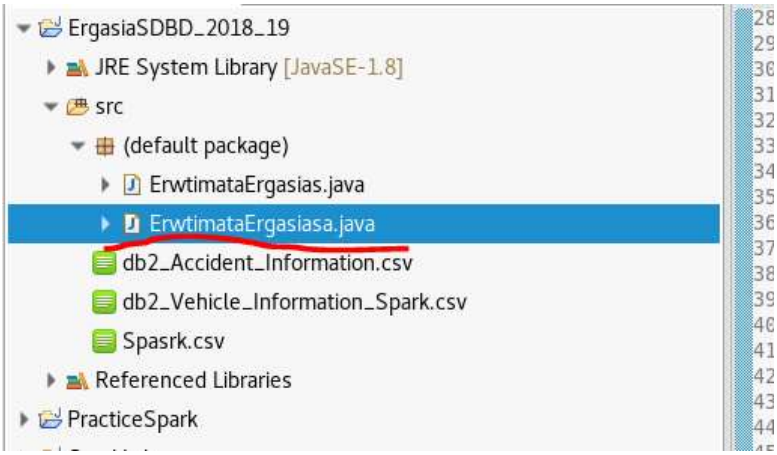
Θα χρειαστώ το δεύτερο αρχείο db2_Vehicle_Information.csv.Και για να βρω πόσοι είναι οι κατασκευαστές των αυτοκινήτων και θα πάρω την στήλη **make**.

	E	F	G	H	I	J	K
_Ve	make	model	Sex_of_Dr	Vehicle_Type			
3	ROVER	45 CLASSIC	Male	109			
	BMW	C1	Male	109			
4	NISSAN	MICRA CEI	Male	109			
	LONDON TAXIS INT	TXII GOLD	Male	109			
1	PIAGGIO	VESPA ET4	Male	Motorcycle 125cc and under			
10	VOLKSWAGEN		Male	109			
	PIAGGIO	VESPA GT	Male	Motorcycle 125cc and under			
	BMW	R1100 RT	Male	109			
3	MERCEDES		Male	109			
4	VOLKSWAGEN	GOLF V5	Female	109			
3	FORD	TRANSIT 3	Male	Van / Goods 3.5 tonnes mgw or under			
2	DENNIS		Male	Bus or coach (17 or more pass seats)			
11	VAUXHALL	CAVALIER	Male	109			
6	DENNIS		Not knowr	Bus or coach (17 or more pass seats)			
1	MERCEDES		Male	Goods 7.5 tonnes mgw and over			
ıt o	MAN		Not knowr	Goods 7.5 tonnes mgw and over			
6	VOLKSWAGEN	GOLF GL A	Male	109			
	PIAGGIO	VESPA ET4	Male	Motorcycle 125cc and under			
6	LONDON TAXIS INT	TX1 BRON	Female	108			
2	BMW	330 CI SPC	Male	109			
9	VOLVO		Male	Van / Goods 3.5 tonnes mgw or under			
5	FIAT	PUNTO 60	Male	109			
3	RENAULT	MEGANE S	Male	109			
13	AUDI	100E AUTO	Male	109			
11	MAZDA	XEDOS 6 S	Not knowr	109			
15	MERCEDES	500 SEL AU	Male	109			
11	ROVER	620 SI AUT	Male	109			
	MAN		Male	Goods 7.5 tonnes mgw and over			

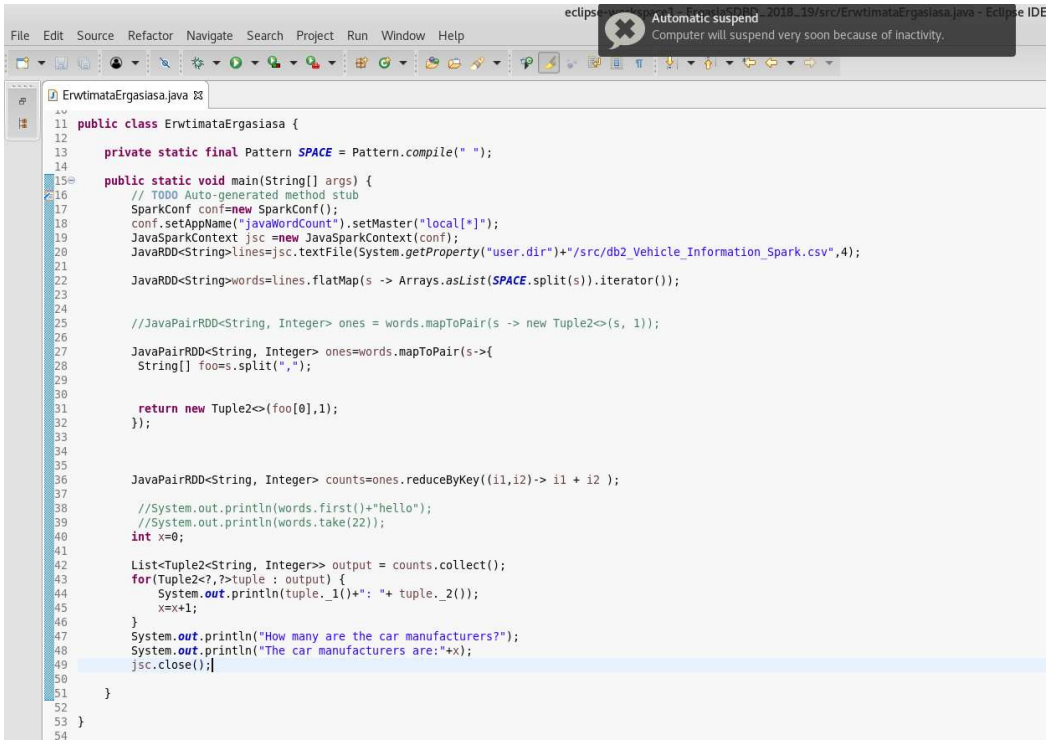
Παίρνω αυτό το αρχείο και το αποθηκεύω στο eclipse μου.



Στην συνέχεια δημιουργώ ένα .java αρχείο με όνομα **ErwtimataErgasiasa.java**



Στην συνέχεια θα κάνω ανάλυση του κώδικα μου.



Αυτό που θέλω να πραγματοποιήσω σε αυτό το ερώτημα είναι να μετρήσω πόσοι είναι οι κατασκευαστές των αυτοκινήτων.

Οπότε αρχικά θα ανοίξω και θα διαβάσω γραμμή γραμμή το αρχείο που έχω **db2_Vehicle_Information_Spark.csv**

Το ανοίγω με την εντολή και διαβάζω το αρχείο μου γραμμή γραμμή:

```
JavaSparkContext jsc =new JavaSparkContext(conf);
JavaRDD<String>lines=jsc.textFile(System.getProperty("user.dir")+"/src/db2_Vehicle_Information_Spark.csv",4);
```

JavaRDD<String>lines=jsc.textFile(System.getProperty(“user.dir”)+”src/db2_Vehicle_Information_Spark.csv”,4);

Στην συνέχεια παίρνω την κάθε γραμμή και την βάζω μεσα σε μια λίστα σαν λέξεις.

```
JavaRDD<String>words=lines.flatMap(s -> Arrays.asList(SPACE.split(s)).iterator());
```

JavaRDD<String>words=lines.flatMap(s→Arrays.asList(SPACE.split(s)).iterator);

Στην συνέχεια παίρνω αυτές τις λέξεις και τις κάνω split με βάση το κόμμα

```
JavaPairRDD<String, Integer> ones=words.mapToPair(s->{
 String[] foo=s.split(",");


 return new Tuple2<>(foo[1],1);
});
```

JavaPairRDD<String, Integer> ones =words.mapToPair(s→{

String[] foo=s.split(“,”)

και επειδή θέλω την πρώτη στήλη παίρνω το foo[0] και το κάνω count

return new Tuple2<>(foo[0],1);

}

Στην συνέχεια κάνω count όλους του κατασκευαστές και τυπώνω πόσοι είναι.

```
        JavaPairRDD<String, Integer> counts=ones.reduceByKey((i1,i2)-> i1 + i2 );

         //System.out.println(words.first()+"hello");
         //System.out.println(words.take(22));
        int x=0;

        List<Tuple2<String, Integer>> output = counts.collect();
        for(Tuple2<?,?>tuple : output) {
            System.out.println(tuple._1()+": "+ tuple._2());
            x=x+1;
        }
        System.out.println("How many are the car manufacturers?");
        System.out.println("The car manufacturers are:"+x);
        jsc.close();

    }
```

Εδώ τυπώνω αν κατασκευαστή πόσες φορές εμφανίστηκε.

System.out.println(“tuple._1()+”:”+tuple._2());

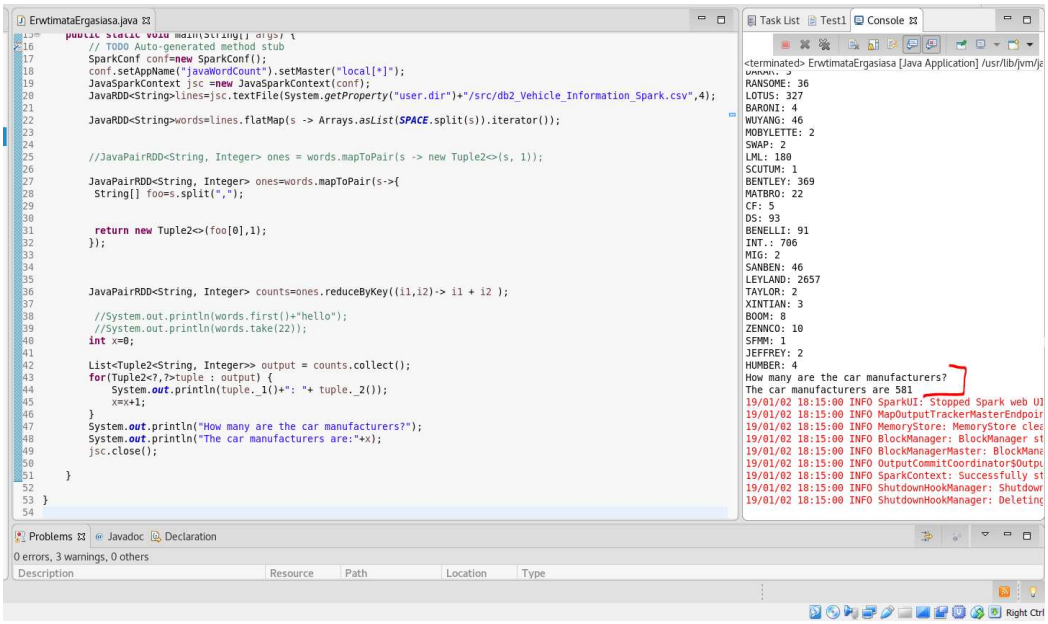
Στο tuple._1() έχω τον κατασκευαστή και στο tuple._2() έχω πόσες φορές εμφανίζεται.

x=x+1;

System.out.println(“How many are the car manufacturers?”);

Εδω τυπώνω ποσοι είναι οι κατασκευαστές.

System.out.println(“The car manufacturers are:”+x);



Όπως βλέπουμε τυπώνονται στην κονσόλα δεξιά πόσες φορές εμφανίζονται οι κατασκευαστές και στο τέλος αφού τους έχουμε μετρήσει με έναν counter **x=x+1;** εμφανίζω πόσοι είναι αυτοί οι κατασκευαστές εδώ είναι.

Οι κατασκευαστές των αυτοκινήτων είναι 581.

(b) Επιλέξτε ένα υποερώτημα από 1.a και προσαρμόστε το σε Spark με τη χρήση των κατάλληλων εντολών που παρέχονται από την πλατφόρμα (π.χ. map, reduce, reducebykey, min, max, κλπ.).

Θα πάρω το υπορερώτημα 1.a

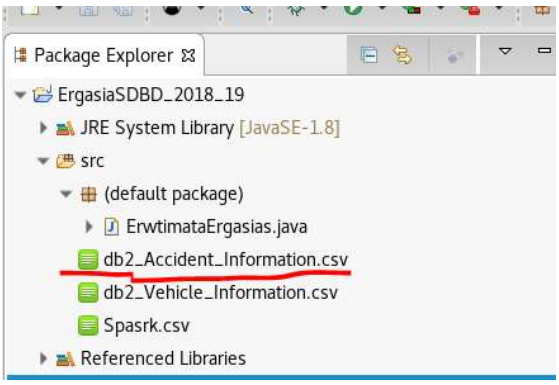
i. Βρείτε πόσα ατυχήματα έγιναν ανά κατηγορία δρόμου (road class).

Για αυτό το ερώτημα θα χρειαστώ από τον πίνακα **db2_Accident_Information.csv** δυο στήλες την στήλη **accident_index** και την στήλη **first_road_class.**

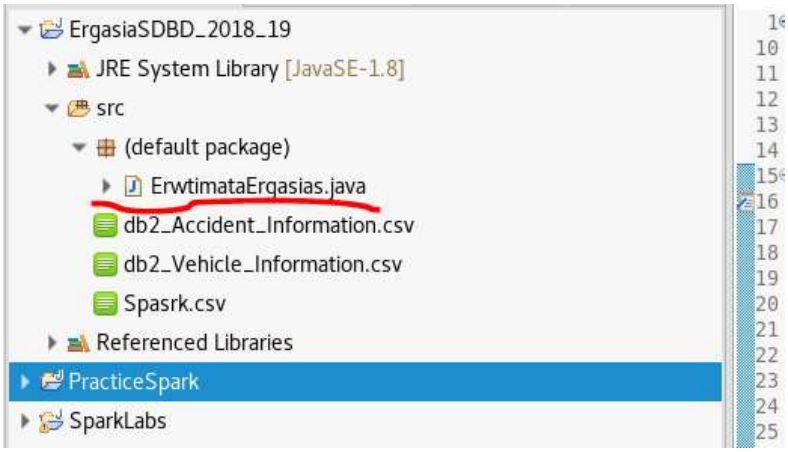
A1

	A	B	C	D
1		Accident_Index	1st_Road_Class	Accident_Severity
2	0	200501BS00001	A	Serious
3	1	200501BS00002	B	Slight
4	2	200501BS00003	C	Slight
5	3	200501BS00004	A	Slight
6	4	200501BS00005	Unclassified	Slight
7	5	200501BS00006	Unclassified	Slight
8	6	200501BS00007	C	Slight
9	7	200501BS00009	A	Slight
10	8	200501BS00010	A	Slight
11	9	200501BS00011	B	Slight
12	10	200501BS00012	A	Slight
13	11	200501BS00014	A	Slight
14	12	200501BS00015	Unclassified	Slight
15	13	200501BS00016	A	Slight
16	14	200501BS00017	A	Slight
17	15	200501BS00018	A	Slight

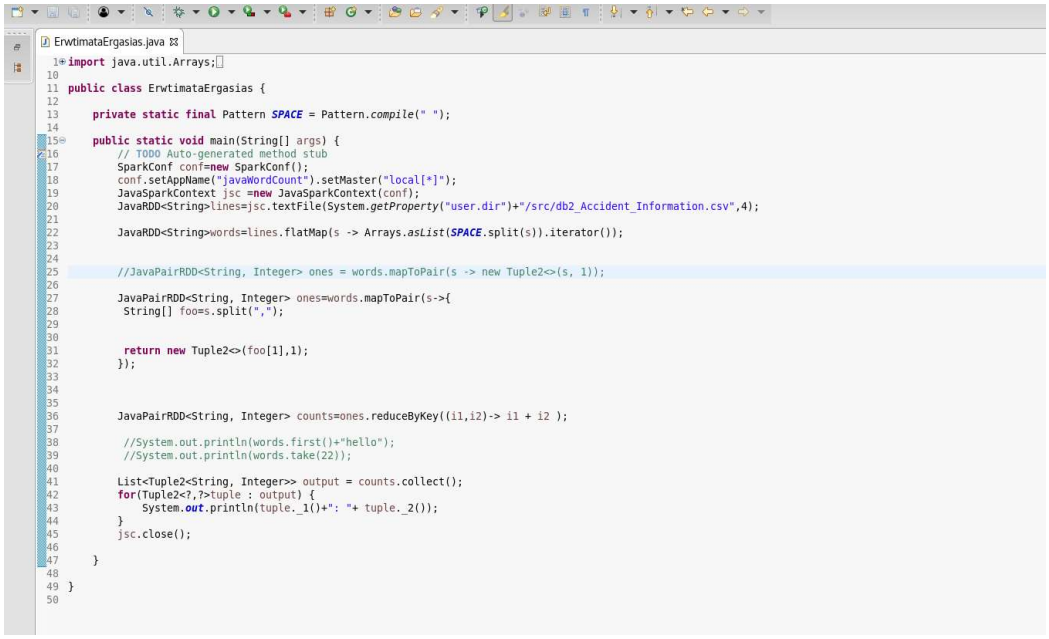
Παίρνω αυτό το αρχείο και το αποθηκεύω στο eclipse μου.



Στην συνέχεια δημιουργώ ενα .java αρχείο με όνομα **ErwtimataErgasias.java**



Στην συνέχεια θα κάνω ανάλυση του κώδικα μου.



```
ErwtimataErgasias.java
import java.util.Arrays;

public class ErwtimataErgasias {

    private static final Pattern SPACE = Pattern.compile(" ");

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        SparkConf conf=new SparkConf();
        conf.setAppName("javaWordCount").setMaster("local[*]");
        JavaSparkContext jsc =new JavaSparkContext(conf);
        JavaRDD<String>lines=jsc.textFile(System.getProperty("user.dir")+"/src/db2_Accident_Information.csv",4);

        JavaRDD<String>words=lines.flatMap(s -> Arrays.asList(SPACE.split(s)).iterator());


        //JavaPairRDD<String, Integer> ones = words.mapToPair(s -> new Tuple2<>(s, 1));

        JavaPairRDD<String, Integer> ones=words.mapToPair(s->{
         String[] foo=s.split(",");


         return new Tuple2<>(foo[1],1);
        });



        JavaPairRDD<String, Integer> counts=ones.reduceByKey((i1,i2)-> i1 + i2 );

         //System.out.println(words.first()+"hello");
         //System.out.println(words.take(22));

        List<Tuple2<String, Integer>> output = counts.collect();
        for(Tuple2<?,?>tuple : output) {
            System.out.println(tuple._1()+": "+ tuple._2());
        }
        jsc.close();

    }
}
```

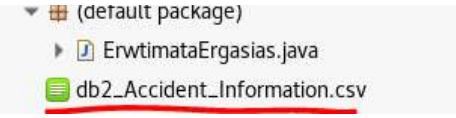
Αυτό που θέλω να πραγματοποιήσω σε αυτό το ερώτημα είναι να μετρήσω την στήλη accident index με βάση την στήλη first_road_class ομαδοποιημένη.

Οπότε αρχικά θα ανοίξω και θα διαβάσω γραμμή γραμμή το αρχείο που έχω db2_Accident_Information.csv

Το ανοίγω με την εντολή και διαβάζω το αρχείο μου γραμμή γραμμή:

```
JavaRDD<String>lines=jsc.textFile(System.getProperty("user.dir")+"/src/db2_Accident_Information.csv",4);
```

JavaRDD<String>lines=jsc.textFile(System.getProperty("user.dir")+"src/db2_Accident_Information.csv",4);



Στην συνέχεια παίρνω την κάθε γραμμή και την βάζω μέσα σε μια λίστα σαν λέξεις.

```
JavaRDD<String>words=lines.flatMap(s -> Arrays.asList(SPACE.split(s)).iterator());
```

JavaRDD<String>words=lines.flatMap(s→Arrays.asList(SPACE.split(s)).iterator);

Στην συνέχεια παίρνω αυτές τις λέξεις και τις κάνω split με βάση το κομμα

```
JavaPairRDD<String, Integer> ones=words.mapToPair(s->{
 String[] foo=s.split(",");


 return new Tuple2<>(foo[1],1);
});
```

JavaPairRDD<String, Integer> ones =words.mapToPair(s→{

String[] foo=s.split(",")

και επειδή θέλω την δεύτερη στήλη παίρνω το foo[1] δηλαδή την στήλη first_road_class και την κάνω count

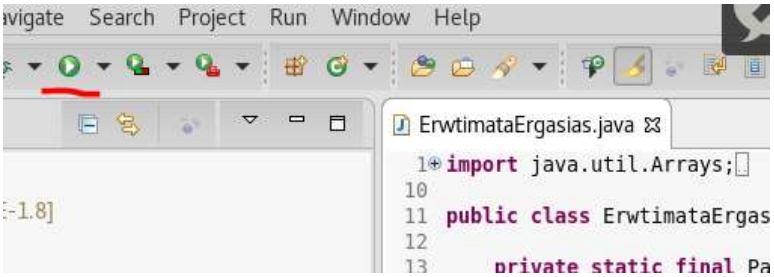
return new Tuple2<>(foo[1],1);

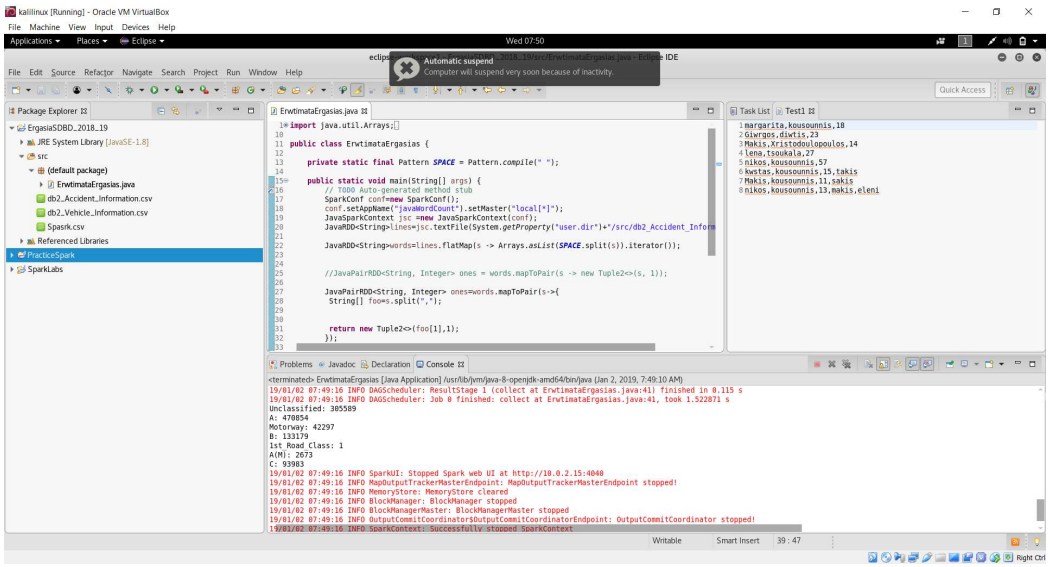
}

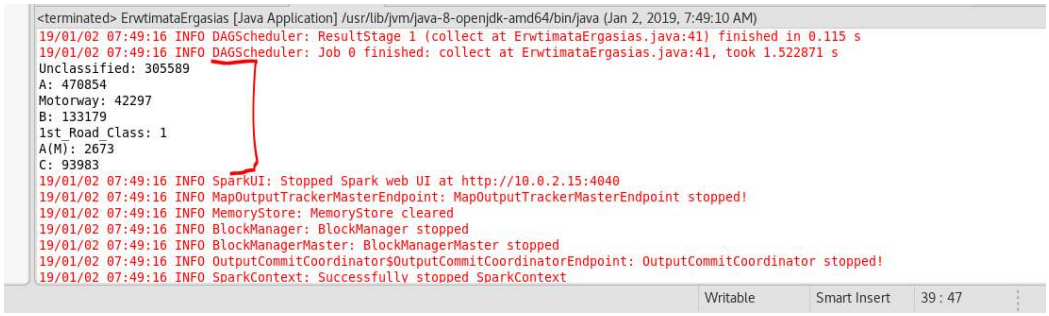
Στην συνέχεια κάνω count τις στήλες με βάση την στήλη first_road_class και τυπώνω τα αποτελέσματα.

```
JavaPairRDD<String, Integer> counts=ones.reduceByKey((i1,i2)-> i1 + i2 );

 //System.out.println(words.first()+"hello");
 //System.out.println(words.take(22));

List<Tuple2<String, Integer>> output = counts.collect();
for(Tuple2<?,?>tuple : output) {
    System.out.println(tuple._1()+": "+ tuple._2());
}
jsc.close();
```

Πατάω play για να τρέξει το πρόγραμμα και βλέπω τα αποτελέσματα.







Βλέπουμε ότι μας εμφανίζει δύο στήλες την στήλη **first_road_class** και την στήλη **count(accident_index).**

Unclassified: 305589

A: 470854

Motorway: 42297

B:133179

1st_Road_class:1

A(M): 2673

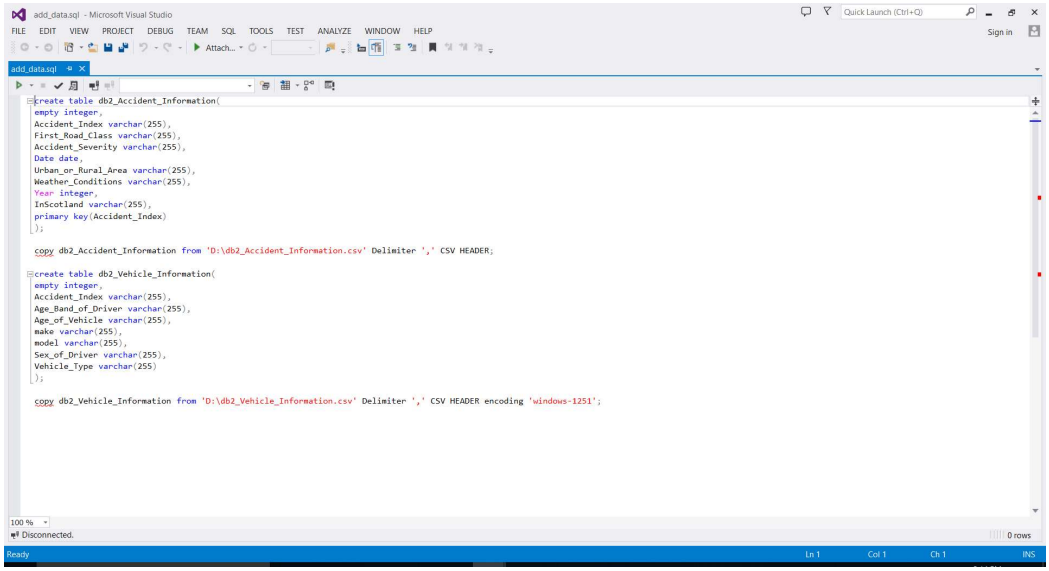
C: 93983

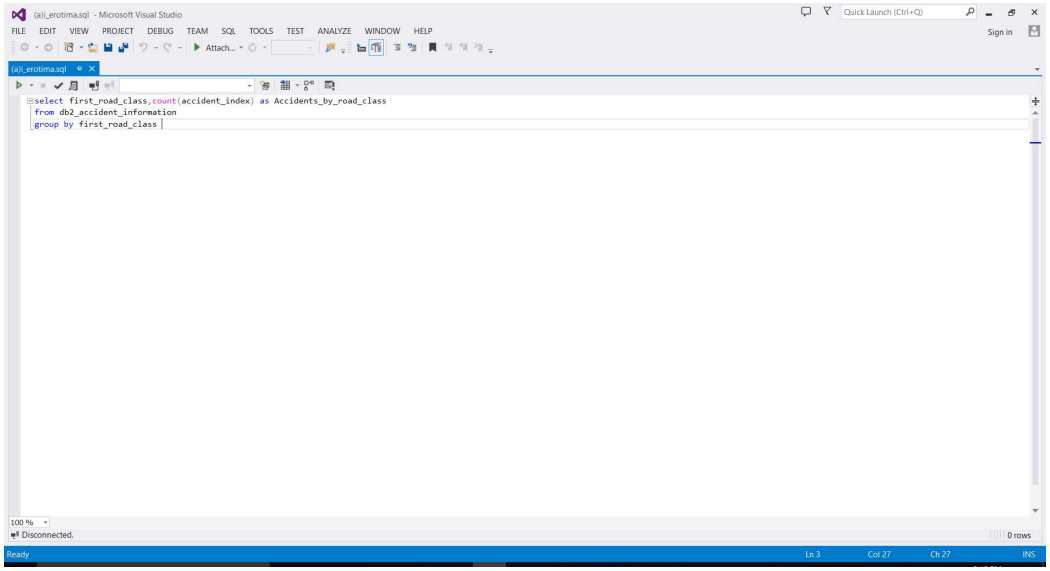
Μας εμφανίζει πόσα ατυχήματα έγιναν ανά κατηγορία δρόμου.

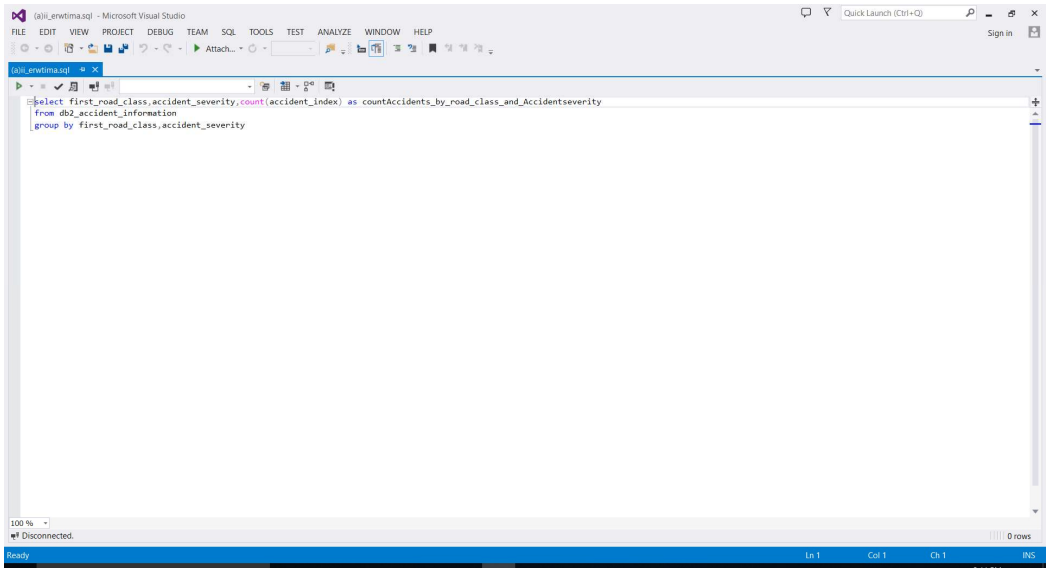
Κώδικας Όλης της Εργασίας

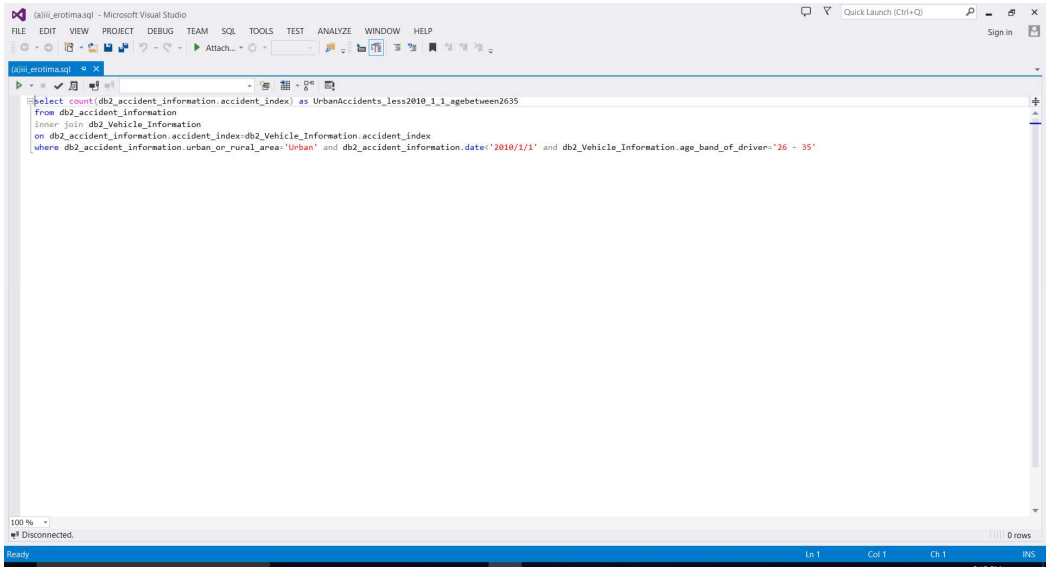
Συστήματα Διαχείρισης Βάσεων Δεδομένων 2018-19

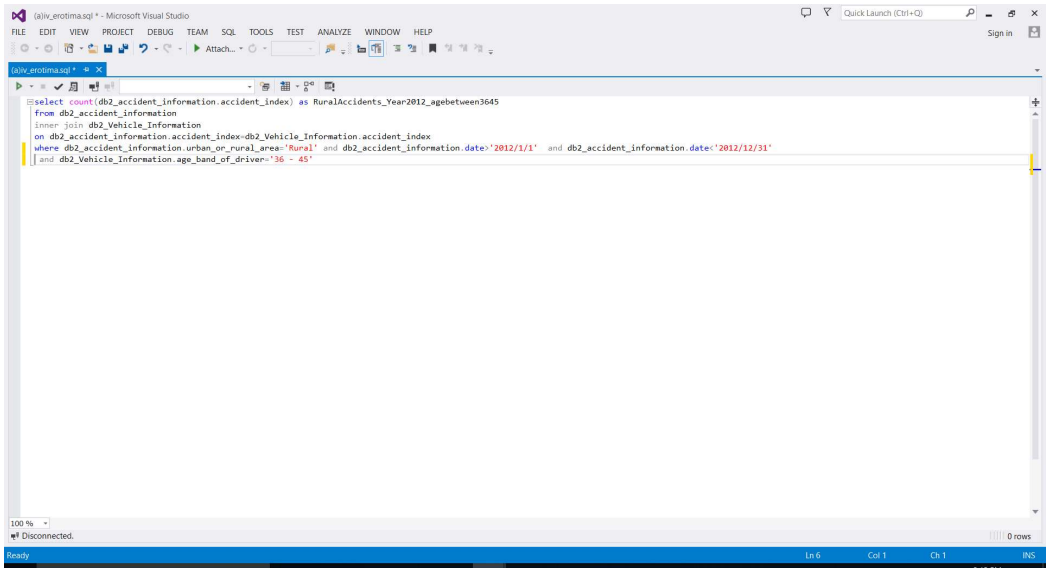
(5ο εξάμηνο)

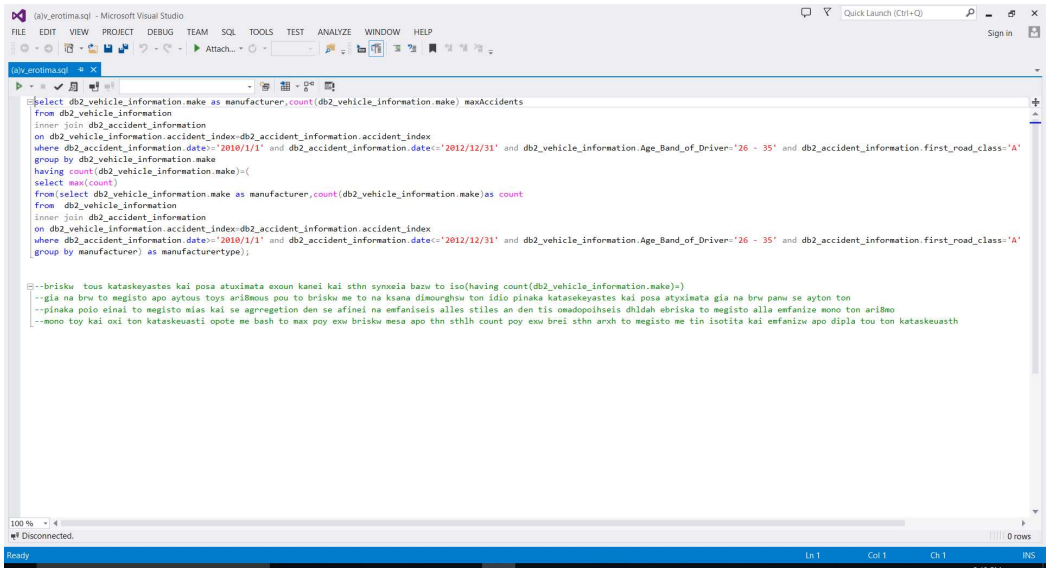


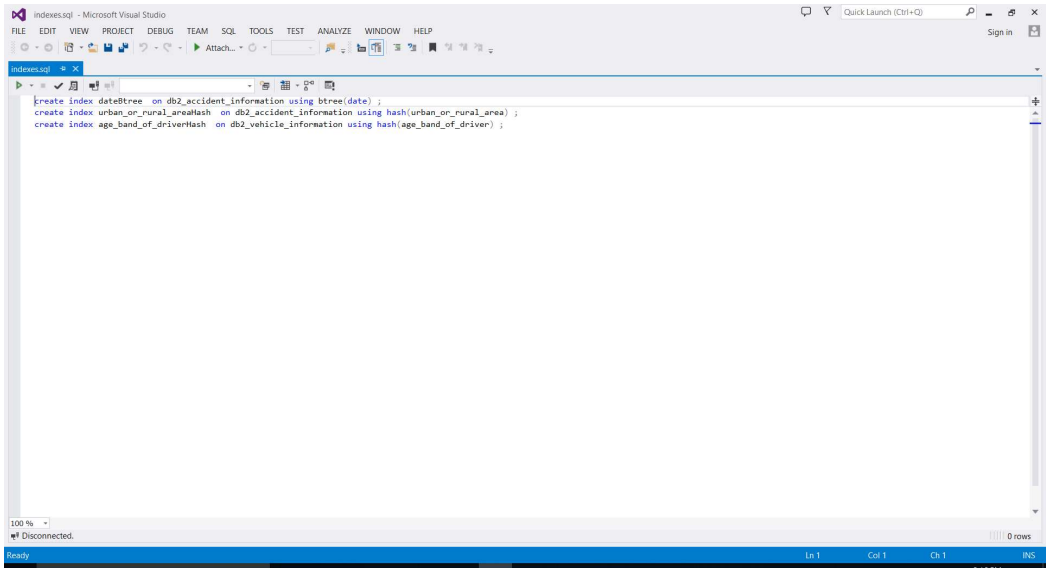

(a)i_erotima.sql - Microsoft Visual Studio
FILE EDIT VIEW PROJECT DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Sign in
Attach...
(a)i_erotima.sql
select first_road_class,count(accident_index) as Accidents_by_road_class
from db2_accident_information
group by first_road_class
100 %
Disconnected.
0 rows
Ready
Ln 3
Col 27
Ch 27
INS

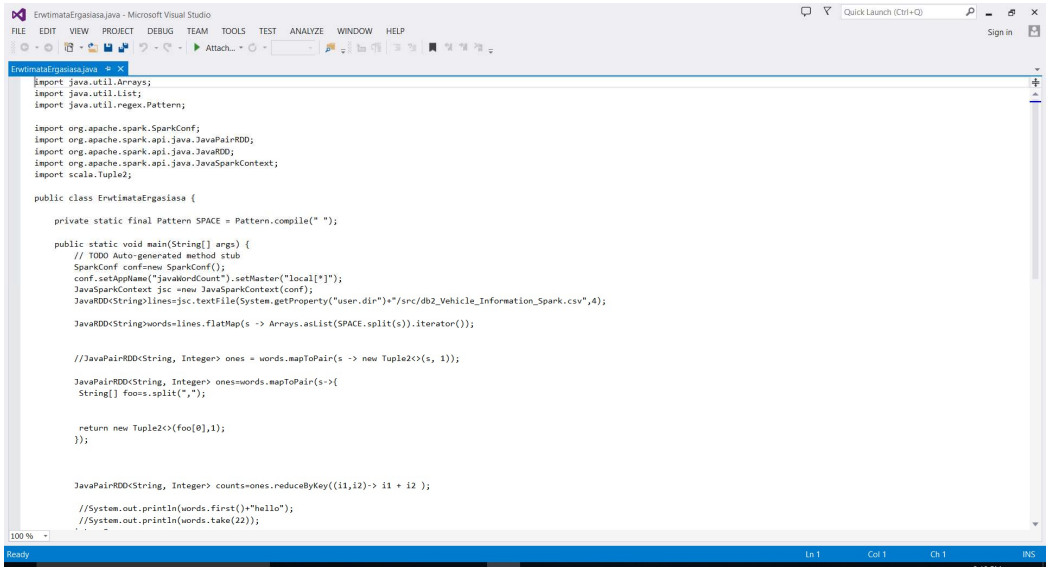

(a)ii_erwtima.sql - Microsoft Visual Studio
FILE EDIT VIEW PROJECT DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Sign in
Attach...
(a)ii_erwtima.sql
select first_road_class,accident_severity,count(accident_index) as countAccidents_by_road_class_and_Accidentseverity
from db2_accident_information
group by first_road_class,accident_severity
100 %
Disconnected.
0 rows
Ready
Ln 1
Col 1
Ch 1
INS

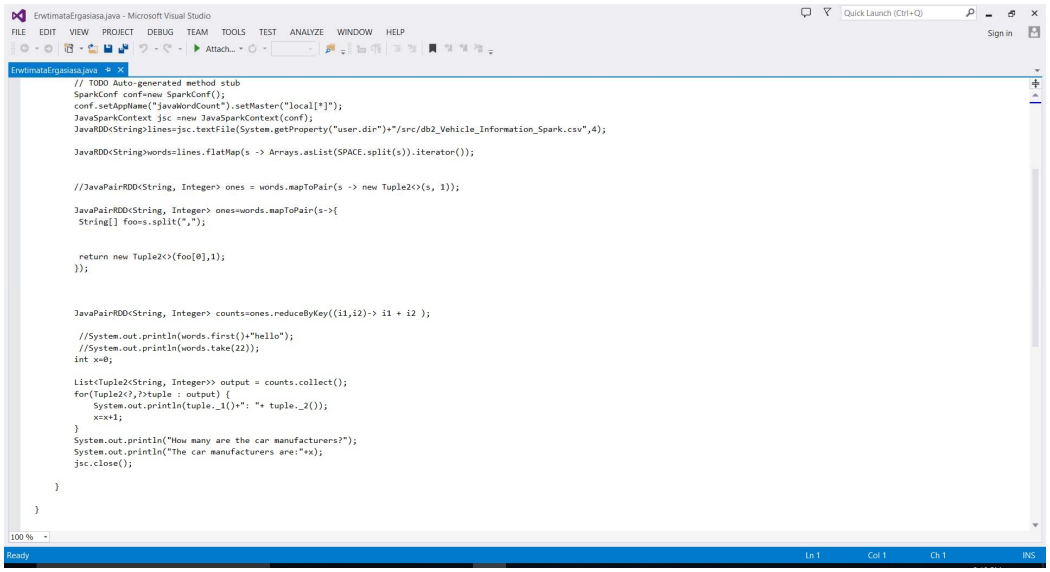

(a)iii_erotima.sql - Microsoft Visual Studio
FILE EDIT VIEW PROJECT DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Sign in
Attach...
(a)iii_erotima.sql
select count(db2_accident_information.accident_index) as UrbanAccidents_less2010_1_1_agebetween2635
from db2_accident_information
inner join db2_Vehicle_Information
on db2_accident_information.accident_index=db2_Vehicle_Information.accident_index
where db2_accident_information.urban_or_rural_area='Urban' and db2_accident_information.date<'2010/1/1' and db2_Vehicle_Information.age_band_of_driver='26 - 35'
100 %
Disconnected.
0 rows
Ready
Ln 1
Col 1
Ch 1
INS

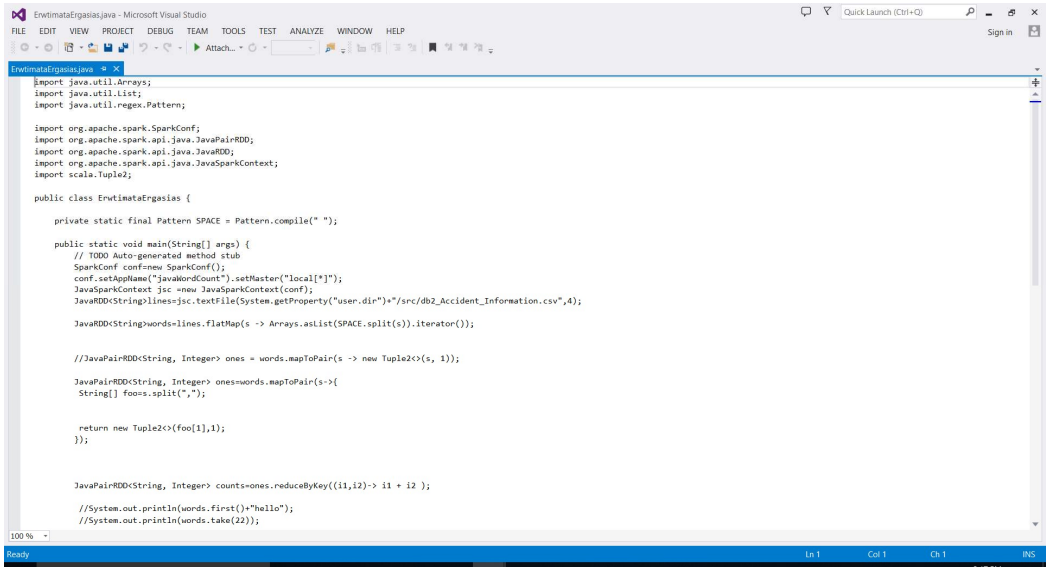

(a)iv_erotima.sql * - Microsoft Visual Studio
FILE EDIT VIEW PROJECT DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Sign in
Attach...
(a)iv_erotima.sql *
select count(db2_accident_information.accident_index) as RuralAccidents_Year2012_agebetween3645
from db2_accident_information
inner join db2_Vehicle_Information
on db2_accident_information.accident_index=db2_Vehicle_Information.accident_index
where db2_accident_information.urban_or_rural_area='Rural' and db2_accident_information.date>'2012/1/1' and db2_accident_information.date<'2012/12/31'
and db2_Vehicle_Information.age_band_of_driver='36 - 45'
100 %
Disconnected.
0 rows
Ready
Ln 6
Col 1
Ch 1
INS

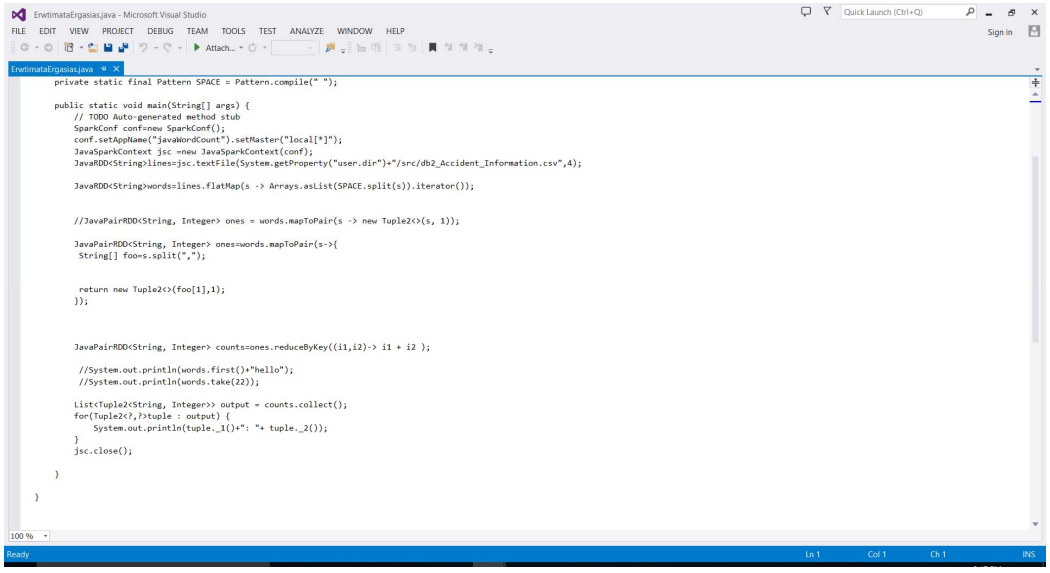

(a)v_erotima.sql - Microsoft Visual Studio
FILE EDIT VIEW PROJECT DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Sign in
Attach...
(a)v_erotima.sql
select db2_vehicle_information.make as manufacturer,count(db2_vehicle_information.make) maxAccidents
from db2_vehicle_information
inner join db2_accident_information
on db2_vehicle_information.accident_index=db2_accident_information.accident_index
where db2_accident_information.date>='2010/1/1' and db2_accident_information.date<='2012/12/31' and db2_vehicle_information.Age_Band_of_Driver='26 - 35' and db2_accident_information.first_road_class='A'
group by db2_vehicle_information.make
having count(db2_vehicle_information.make)=(
select max(count)
from(select db2_vehicle_information.make as manufacturer,count(db2_vehicle_information.make)as count
from db2_vehicle_information
inner join db2_accident_information
on db2_vehicle_information.accident_index=db2_accident_information.accident_index
where db2_accident_information.date>='2010/1/1' and db2_accident_information.date<='2012/12/31' and db2_vehicle_information.Age_Band_of_Driver='26 - 35' and db2_accident_information.first_road_class='A'
group by manufacturer) as manufacturertype);
--briskw tous kataskeyastes kai posa atuximata exoun kanei kai sthn synxeia bazw to iso(having count(db2_vehicle_information.make)=)
--gia na brw to megisto apo aytous toys ari8mous pou to briskw me to na ksana dimourghsw ton idio pinaka katasekeyastes kai posa atyximata gia na brw panw se ayton ton
--pinaka poio einai to megisto mias kai se agrregetion den se afinei na emfaniseis alles stiles an den tis omadopoihseis dhldah ebriska to megisto alla emfanize mono ton ari8mo
--mono toy kai oxi ton kataskeuasti opote me bash to max poy exw briskw mesa apo thn sthlh count poy exw brei sthn arxh to megisto me tin isotita kai emfanizw apo dipla tou ton kataskeuasth
100 %
Disconnected.
0 rows
Ready
Ln 1
Col 1
Ch 1
INS


indexes.sql - Microsoft Visual Studio
FILE EDIT VIEW PROJECT DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Sign in
Attach...
indexes.sql
create index date8tree on db2_accident_information using btree(date) ;
create index urban_or_rural_areaHash on db2_accident_information using hash(urban_or_rural_area) ;
create index age_band_of_driverHash on db2_vehicle_information using hash(age_band_of_driver) ;
100 %
Disconnected.
0 rows
Ready
Ln 1
Col 1
Ch 1
INS


ErwtimataErgasiasa.java - Microsoft Visual Studio
FILE EDIT VIEW PROJECT DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Sign in
Attach...
ErwtimataErgasiasa.java
import java.util.Arrays;
import java.util.List;
import java.util.regex.Pattern;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import scala.Tuple2;
public class ErwtimataErgasiasa {
private static final Pattern SPACE = Pattern.compile(" ");
public static void main(String[] args) {
// TODO Auto-generated method stub
SparkConf conf=new SparkConf();
conf.setAppName("javaWordCount").setMaster("local[*]");
JavaSparkContext jsc =new JavaSparkContext(conf);
JavaRDD<String>lines=jsc.textFile(System.getProperty("user.dir")+"/src/db2_Vehicle_Information_Spark.csv",4);
JavaRDD<String>words=lines.flatMap(s -> Arrays.asList(SPACE.split(s)).iterator());
//JavaPairRDD<String, Integer> ones = words.mapToPair(s -> new Tuple2<>(s, 1));
JavaPairRDD<String, Integer> ones=words.mapToPair(s->{
String[] foo=s.split(",");
return new Tuple2<>(foo[0],1);
});
JavaPairRDD<String, Integer> counts=ones.reduceByKey((i1,i2)-> i1 + i2);
//System.out.println(words.first()+"hello");
//System.out.println(words.take(22));
100 %
Ready
Ln 1
Col 1
Ch 1
INS


ErwtimataErgasiasa.java - Microsoft Visual Studio
FILE EDIT VIEW PROJECT DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Sign in
Attach...
ErwtimataErgasiasa.java
// TODO Auto-generated method stub
SparkConf conf=new SparkConf();
conf.setAppName("javaWordCount").setMaster("local[*]");
JavaSparkContext jsc =new JavaSparkContext(conf);
JavaRDD<String>lines=jsc.textFile(System.getProperty("user.dir")+"/src/db2_Vehicle_Information_Spark.csv",4);
JavaRDD<String>words=lines.flatMap(s -> Arrays.asList(SPACE.split(s)).iterator());
//JavaPairRDD<String, Integer> ones = words.mapToPair(s -> new Tuple2<>(s, 1));
JavaPairRDD<String, Integer> ones=words.mapToPair(s->{
String[] foo=s.split(",");
return new Tuple2<>(foo[0],1);
});
JavaPairRDD<String, Integer> counts=ones.reduceByKey((i1,i2)-> i1 + i2);
//System.out.println(words.first()+"hello");
//System.out.println(words.take(22));
int x=0;
List<Tuple2<String, Integer>> output = counts.collect();
for(Tuple2<?,?>tuple : output) {
System.out.println(tuple._1()+": "+ tuple._2());
x=x+1;
}
System.out.println("How many are the car manufacturers?");
System.out.println("The car manufacturers are:"+x);
jsc.close();
}
}
100 %
Ready
Ln 1
Col 1
Ch 1
INS


ErwtimataErgasias.java - Microsoft Visual Studio
FILE EDIT VIEW PROJECT DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Sign in
Attach...
ErwtimataErgasias.java
import java.util.Arrays;
import java.util.List;
import java.util.regex.Pattern;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import scala.Tuple2;
public class ErwtimataErgasias {
private static final Pattern SPACE = Pattern.compile(" ");
public static void main(String[] args) {
// TODO Auto-generated method stub
SparkConf conf=new SparkConf();
conf.setAppName("javaWordCount").setMaster("local[*]");
JavaSparkContext jsc =new JavaSparkContext(conf);
JavaRDD<String>lines=jsc.textFile(System.getProperty("user.dir")+"/src/db2_Accident_Information.csv",4);
JavaRDD<String>words=lines.flatMap(s -> Arrays.asList(SPACE.split(s)).iterator());
//JavaPairRDD<String, Integer> ones = words.mapToPair(s -> new Tuple2<>(s, 1));
JavaPairRDD<String, Integer> ones=words.mapToPair(s->{
String[] foo=s.split(",");
return new Tuple2<>(foo[1],1);
});
JavaPairRDD<String, Integer> counts=ones.reduceByKey((i1,i2)-> i1 + i2);
//System.out.println(words.first()+"hello");
//System.out.println(words.take(22));
100 %
Ready
Ln 1
Col 1
Ch 1
INS


ErwtimataErgasias.java - Microsoft Visual Studio
FILE EDIT VIEW PROJECT DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Sign in
Attach...
ErwtimataErgasias.java
private static final Pattern SPACE = Pattern.compile(" ");
public static void main(String[] args) {
// TODO Auto-generated method stub
SparkConf conf=new SparkConf();
conf.setAppName("javaWordCount").setMaster("local[*]");
JavaSparkContext jsc =new JavaSparkContext(conf);
JavaRDD<String>lines=jsc.textFile(System.getProperty("user.dir")+"/src/db2_Accident_Information.csv",4);
JavaRDD<String>words=lines.flatMap(s -> Arrays.asList(SPACE.split(s)).iterator());
//JavaPairRDD<String, Integer> ones = words.mapToPair(s -> new Tuple2<>(s, 1));
JavaPairRDD<String, Integer> ones=words.mapToPair(s->{
String[] foo=s.split(",");
return new Tuple2<>(foo[1],1);
});
JavaPairRDD<String, Integer> counts=ones.reduceByKey((i1,i2)-> i1 + i2);
//System.out.println(words.first()+"hello");
//System.out.println(words.take(22));
List<Tuple2<String, Integer>> output = counts.collect();
for(Tuple2<?,?>tuple : output) {
System.out.println(tuple._1()+": "+ tuple._2());
}
jsc.close();
}
}
100 %
Ready
Ln 1
Col 1
Ch 1
INS