

Μάθημα: «Συστήματα Διαχείρισης Βάσεων Δεδομένων (5^ο εξ.)»

Ομάδα Εργασίας:

ΑΜ : Π15128, Σκάρος Γεώργιος
ΑΜ : Π15119, Ποδηματάς Δημήτριος
ΑΜ : Π15123, Ιωσήφ Σαγιέντ

Ημερομηνία παράδοσης : 31/01/2018



Μέρος 1^ο

Πριν την εκτέλεση των ερωτημάτων του πρώτου μέρους έπρεπε να φτιάξουμε 1 πίνακα στην βάση δεδομένων που φτιάχαμε για να μπορέσουμε να του περάσουμε το dataset. Οι εντολές που χρησιμοποιήθηκαν για τη δημιουργία του πίνακα (Create table) και την εισαγωγή των δεδομένων σε αυτόν είναι τα εξής :

Δημιουργία πίνακα

```
CREATE TABLE mainTable (
    id varchar,
    times timestamp,
    distance double precision,
    region_id smallint
);
```

Εισαγωγή δεδομένων

```
COPY mainTable(id, times, distance, region_id)
FROM 'C:\Desktop\db2_project_data.zip'
WITH CSV HEADER;
```

Ανανέωση στατιστικών

```
VACUUM FULL mainTable;
```

The screenshot shows the IBM Data Studio interface with a query editor window open. The query editor contains the following SQL code:

```
CREATE TABLE mainTable (
    id varchar,
    times timestamp,
    distance double precision,
    region_id smallint
);
copy mainTable FROM 'C:\Users\George\Desktop\db2_project_data.zip' DELIMITER ',' CSV HEADER;
select * from mainTable;
```

Below the query editor is an output pane displaying the results of the query. The results are a table with 27 rows, showing data from the mainTable. The columns are:

	id character varying	times timestamp without time zone	distance double precision	region_id smallint
1	799512-CLEND-304-305-Kupixd-07	2017-01-29 12:18:28	316.450197132761	41
2	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:18:28	19578.1203565881	34
3	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:19:28	122.716701516407	34
4	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:20:28	282.000211404204	7
5	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:20:28	236.224411696094	50
6	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:21:28	181.545385902551	45
7	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:22:28	221.539417753607	40
8	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:23:28	371.58695241235	4
9	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:25:28	459.907666130108	23
10	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:26:28	311.955768071551	7
11	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:27:28	394.356752419485	43
12	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:28:28	261.423153992507	44
13	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:30:28	237.934320532999	9
14	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:31:28	679.796101734798	22
15	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:33:28	531.570304325394	21
16	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:33:28	351.790560487048	3
17	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:34:28	185.30467931303	47
18	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:35:28	169.714177504138	30
19	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:35:28	211.982294326504	34
20	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:36:28	412.646404727088	30
21	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:36:28	186.236681099172	43
22	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:37:28	677.899103603851	50
23	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:38:28	589.651266227327	44
24	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:39:28	426.443744514835	22
25	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:39:28	550.409931139568	20
26	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:40:28	546.979220925371	41
27	799512-CLEND-304-305-Kupixd-07	2017-02-05 11:41:28	261.556024859923	42

- A. ερώτημα

Τα queries του Α ερωτήματος είναι τα παρακάτω:

I.

```

SELECT A.id , SUM(A.distance) AS MaxDistance
FROM(
    SELECT id , times ,distance
    FROM maintable
    WHERE times < '2017-01-03 03:58:01'
) AS A
GROUP BY A.id
ORDER BY MaxDistance DESC
LIMIT 1;

```

The screenshot shows the IBM DB2 Query Editor interface. The SQL Editor pane contains the query code provided above. The Data Output pane shows the results of the query execution:

	id character varying	maxdistance double precision
1	9741601-CALEND-304-305-Καθημερινή-11	1160454.31035527

At the bottom right of the interface, there is a status bar with the text "OK.", "DOS Ln 9, Col 9, Ch 217", "1 row.", and "1.1 secs". An arrow points to the "1.1 secs" text, indicating the execution time of the query.

II. SELECT AVG(distance) AS AverageDistance
 FROM maintable
 WHERE times >
 (select max(times) AS now
 from maintable
) -interval'1 month';

The screenshot shows a DB2 SQL editor window titled "Query - db2 on postgres@localhost:5432 *". The window has several panes:

- SQL Editor:** Contains the query code from above.
- Scratch pad:** Empty.
- Output pane:** Contains the results of the query execution.
- Bottom status bar:** Shows "OK.", "DOS", "Ln 7, Col 30, Ch 152", "1 row.", and "2.5 secs". A large black arrow points downwards from the status bar towards the bottom right corner.

The output pane displays the following result:

	averagedistance
1	566.045178187881

```

III.   SELECT id, to_char(times , 'Mon'), sum(distance)
      FROM maintable
      GROUP BY id , to_char(times , 'Mon');

```

The screenshot shows a DB2 Query tool interface with the following details:

- Title Bar:** Query - db2 on postgres@localhost:5432 *
- Menu Bar:** File Edit Query Favourites Macros View Help
- Toolbar:** Includes icons for file operations, search, and database management.
- SQL Editor:** Contains the SQL query:

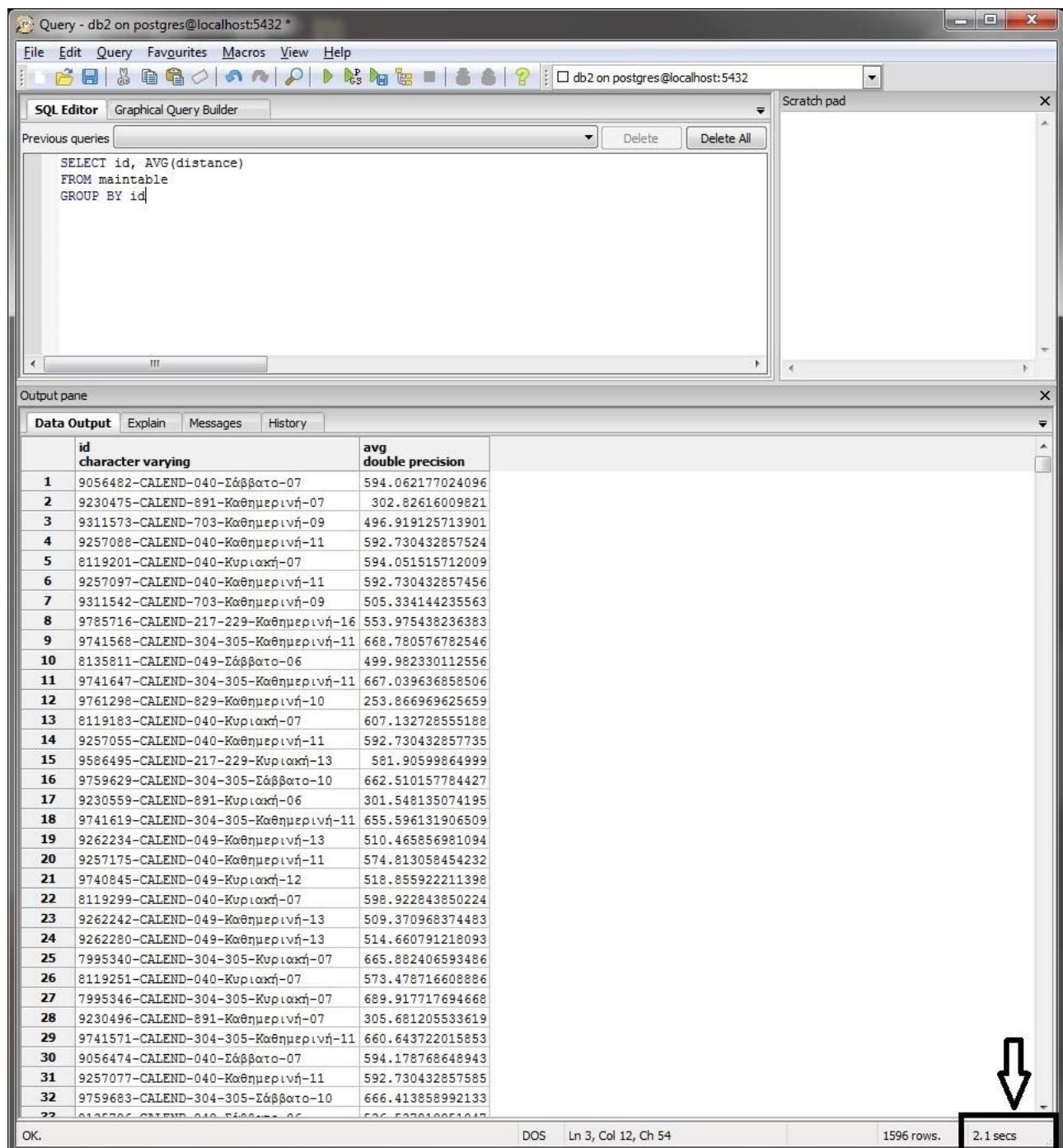
```

SELECT id, to_char(times , 'Mon'), sum(distance)
FROM maintable
GROUP BY id , to_char(times , 'Mon');

```
- Scratch pad:** An empty pane.
- Output pane:** Shows the results of the query in a Data Output tabular format. The table has columns: id (character varying), to_char (text), and sum (double precision). The data consists of 11155 rows, took 55.9 secs, and spans pages 304-305. The first few rows are:

	id	to_char	sum
	character varying	text	double precision
1	7079824-CALEND-703-Kuprioxn-04	Apr	78795.2098646022
2	7079824-CALEND-703-Kuprioxn-04	Mar	11152.1631362318
3	7079824-CALEND-703-Kuprioxn-04	May	80993.3605512418
4	7079825-CALEND-703-Kuprioxn-04	Apr	111635.273165609
5	7079825-CALEND-703-Kuprioxn-04	Mar	35464.9932319005
6	7079825-CALEND-703-Kuprioxn-04	May	138116.47327556
7	7079826-CALEND-703-Kuprioxn-04	Apr	111495.270758132
8	7079826-CALEND-703-Kuprioxn-04	Mar	89977.0949399378
9	7079826-CALEND-703-Kuprioxn-04	May	136826.813649016
10	7079827-CALEND-703-Kuprioxn-04	Apr	111083.586123057
11	7079827-CALEND-703-Kuprioxn-04	Mar	100691.735183697
12	7079827-CALEND-703-Kuprioxn-04	May	139580.017398549
13	7079828-CALEND-703-Kuprioxn-04	Apr	112042.33495432
14	7079828-CALEND-703-Kuprioxn-04	Mar	101335.8228801293
15	7079828-CALEND-703-Kuprioxn-04	May	138158.869598872
16	7079846-CALEND-703-Kuprioxn-04	Apr	55876.6884706819
17	7079846-CALEND-703-Kuprioxn-04	Mar	6543.36632043179
18	7079846-CALEND-703-Kuprioxn-04	May	56890.3634257311
19	7079847-CALEND-703-Kuprioxn-04	Apr	105962.851107804
20	7079847-CALEND-703-Kuprioxn-04	Mar	93345.3005943899
21	7079847-CALEND-703-Kuprioxn-04	May	132080.428182452
22	7079848-CALEND-703-Kuprioxn-04	Apr	105813.317941618
23	7079848-CALEND-703-Kuprioxn-04	Mar	93131.0028740626
24	7079848-CALEND-703-Kuprioxn-04	May	129974.615200226
25	7079875-CALEND-703-Kuprioxn-04	Apr	116559.819966464
26	7079875-CALEND-703-Kuprioxn-04	Mar	14731.853011408
27	7079875-CALEND-703-Kuprioxn-04	May	119970.713442294
28	7079878-CALEND-703-Kuprioxn-04	Apr	105111.266426146
29	7079878-CALEND-703-Kuprioxn-04	Mar	57544.7674154455
30	7079878-CALEND-703-Kuprioxn-04	May	131174.840402934
31	7079879-CALEND-703-Kuprioxn-04	Apr	0.00278171513814659
32	7079879-CALEND-703-Kuprioxn-04	May	0.00221731004931778
- Status Bar:** OK. DOS Ln 3, Col 37, Ch 104 11155 rows. 55.9 secs

IV. SELECT id, AVG(distance)
 FROM maintable
 GROUP BY id;



```
Query - db2 on postgres@localhost:5432 *
File Edit Query Favourites Macros View Help
SQL Editor Graphical Query Builder
Previous queries Delete Delete All
SELECT id, AVG(distance)
FROM maintable
GROUP BY id

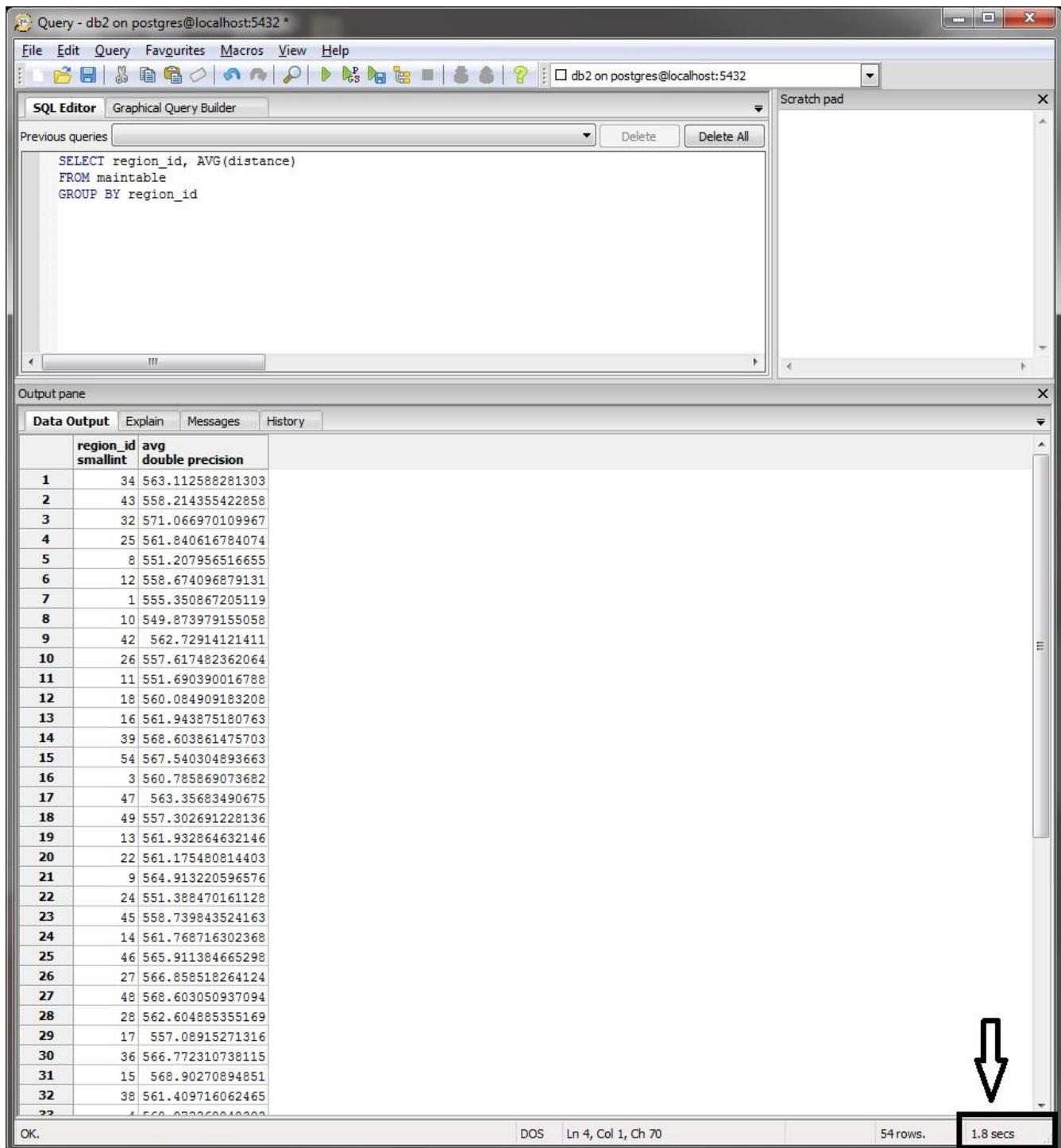
Output pane
Data Output Explain Messages History

```

	id character varying	avg double precision
1	9056482-CALEND-040-Σάββατο-07	594.062177024096
2	9230475-CALEND-891-Καθημερινή-07	302.82616009821
3	9311573-CALEND-703-Καθημερινή-09	496.919125713901
4	9257088-CALEND-040-Καθημερινή-11	592.730432857524
5	8119201-CALEND-040-Κυριακή-07	594.051515712009
6	9257097-CALEND-040-Καθημερινή-11	592.730432857456
7	9311542-CALEND-703-Καθημερινή-09	505.334144235563
8	9785716-CALEND-217-229-Καθημερινή-16	553.975438236383
9	9741568-CALEND-304-305-Καθημερινή-11	668.780576782546
10	8135811-CALEND-049-Σάββατο-06	499.982330112556
11	9741647-CALEND-304-305-Καθημερινή-11	667.039636858506
12	9761298-CALEND-829-Καθημερινή-10	253.866969625659
13	8119183-CALEND-040-Κυριακή-07	607.132728555188
14	9257055-CALEND-040-Καθημερινή-11	592.730432857735
15	9586495-CALEND-217-229-Κυριακή-13	581.90599864999
16	9759629-CALEND-304-305-Σάββατο-10	662.510157784427
17	9230559-CALEND-891-Κυριακή-06	301.548135074195
18	9741619-CALEND-304-305-Καθημερινή-11	655.596131906509
19	9262234-CALEND-049-Καθημερινή-13	510.465856981094
20	9257175-CALEND-040-Καθημερινή-11	574.813058454232
21	9740845-CALEND-049-Κυριακή-12	518.855922211398
22	8119299-CALEND-040-Κυριακή-07	598.922843850224
23	9262242-CALEND-049-Καθημερινή-13	509.370968374483
24	9262280-CALEND-049-Καθημερινή-13	514.660791218093
25	7995340-CALEND-304-305-Κυριακή-07	665.882406593486
26	8119251-CALEND-040-Κυριακή-07	573.478716608886
27	7995346-CALEND-304-305-Κυριακή-07	689.917717694668
28	9230496-CALEND-891-Καθημερινή-07	305.681205533619
29	9741571-CALEND-304-305-Καθημερινή-11	660.643722015853
30	9056474-CALEND-040-Σάββατο-07	594.178768648943
31	9257077-CALEND-040-Καθημερινή-11	592.730432857585
32	9759683-CALEND-304-305-Σάββατο-10	666.413858992133
33	9135706-CALEND-040-Σάββατο-06	595.522010051047

OK. DOS Ln 3, Col 12, Ch 54 1596 rows. 2.1 secs

V. SELECT region_id, AVG(distance)
 FROM maintable
 GROUP BY region_id;



The screenshot shows a DB2 query tool interface with the following details:

- Title Bar:** "Query - db2 on postgres@localhost:5432 *"
- Menu Bar:** File, Edit, Query, Favourites, Macros, View, Help
- Toolbar:** Includes icons for file operations, search, and database management.
- SQL Editor:** Shows the query: "SELECT region_id, AVG(distance) FROM maintable GROUP BY region_id".
- Scratch pad:** An empty pane.
- Output pane:**
 - Data Output:** Selected tab. Shows a table with two columns: "region_id" (smallint) and "avg" (double precision).
 - Table Data:**

region_id	avg
1	563.112588281303
2	558.214355422858
3	571.066970109967
4	561.840616784074
5	551.207956516655
6	558.674096879131
7	555.350867205119
8	549.873979155058
9	562.72914121411
10	557.617482362064
11	551.690390016788
12	560.084909183208
13	561.943875180763
14	568.603861475703
15	567.540304893663
16	560.785869073682
17	563.35683490675
18	557.302691228136
19	561.932864632146
20	561.175480814403
21	564.913220596576
22	551.388470161128
23	558.739843524163
24	561.768716302368
25	565.911384665298
26	566.858518264124
27	568.603050937094
28	562.604885355169
29	557.08915271316
30	566.772310738115
31	568.90270894851
32	561.409716062465
33	560.072200000000
 - Bottom Status Bar:** OK., DOS, Ln 4, Col 1, Ch 70, 54 rows, 1.8 secs.

- **B. Ερώτημα**

Για το β ερώτημα χρειάζεται αρχικά να δώσουμε περισσότερη μνήμη στους buffers με την εντολή ALTER SYSTEM SET shared_buffers TO '1GB';. Έπειτα χρειάζεται να κάνουμε επανεκκίνηση του PostgreSQL server με την εντολή SELECT pg_reload_conf();. Στην συνέχεια εκτέλεσα τα queries του προηγούμενου ερωτήματος. Σε αυτό το σημείο είναι σημαντικό να επισημάνω ότι στον υπολογιστή μου, στο σπίτι όπου εκτελούσα τα queries έχω ssd οπότε σε αυτό το ερώτημα δεν θα δούμε μεγάλες διάφορες στους χρόνους. Παρόλα αυτά έχω συμπεριλάβει στα screenshot που έχω κάνει και αυτά της κανονικής εκτέλεσης αλλά και εκείνα του explain.

i.

The screenshot displays two windows of the db2 client interface. The top window is titled "Query - db2 on postgres@localhost:5432" and contains the following SQL code:

```

ALTER SYSTEM SET shared_buffers TO '1GB';
SELECT pg_reload_conf()

SELECT A.id , SUM(A.distance) AS MaxDistance
FROM(
    SELECT id , times ,distance
    FROM maintable
    WHERE times < '2017-01-03 03:58:01'
) AS A
GROUP BY A.id
ORDER BY MaxDistance DESC
LIMIT 1;
  
```

The output pane shows a single row of data:

	id character varying	maxdistance double precision
1	9741601-CALEND-304-305-Καθημερινή-11	1160454.31035527

The bottom window is also titled "Query - db2 on postgres@localhost:5432" and shows the EXPLAIN PLAN for the same query:

```

explain(SELECT A.id , SUM(A.distance) AS MaxDistance
FROM(
    SELECT id , times ,distance
    FROM maintable
    WHERE times < '2017-01-03 03:58:01'
) AS A
GROUP BY A.id
ORDER BY MaxDistance DESC
LIMIT 1;
  
```

The output pane displays the query plan with numbered steps:

QUERY PLAN
text

- 1 Limit (cost=158180.43..158180.43 rows=1 width=51)
- 2 -> Sort (cost=158180.43..158181.03 rows=240 width=51)
- 3 Sort Key: (sum(maintable.distance)) DESC
- 4 -> HashAggregate (cost=158176.83..158179.23 rows=240 width=51)
- 5 Group Key: maintable.id
- 6 -> Seq Scan on maintable (cost=0.00..153250.20 rows=985325 width=51)
- 7 Filter: (times < '2017-01-03 03:58:01'::timestamp without time zone)

At the bottom of the client interface, there are status bars indicating "OK.", "DOS Ln 13, Col 10, Ch 301", "7 rows.", and "15 msec".

ii.

The image shows two side-by-side screenshots of a DB2 graphical interface. Both windows have the title "Query - db2 on postgres@localhost:5432".

Top Window:

- SQL Editor:** Contains the following SQL code:

```
ALTER SYSTEM SET shared_buffers TO '1GB';
SELECT pg_reload_conf()

SELECT AVG(distance) AS AverageDistance
FROM maintable
WHERE times >
      (           select max(times) AS now
        from maintable
      ) -interval'1 month'
```
- Output pane:** Shows the result of the query in a table:

averagedistance double precision
1 566.045178187881

Bottom Window:

- SQL Editor:** Contains the same SQL code as the top window.
- Output pane:** Shows the result of the query in a table:

averagedistance double precision
1 566.045178187881

Both windows have a status bar at the bottom with the following information:

- OK.
- DOS Ln 11, Col 1, Ch 230
- 1 row. 2.5 secs

iii.

The image displays two side-by-side screenshots of DB2 query tools, both titled "Query - db2 on postgres@localhost:5432".

Top Window:

- SQL Editor:** Contains the following SQL code:

```
ALTER SYSTEM SET shared_buffers TO '1GB';
SELECT pg_reload_conf()
      SELECT id, to_char(times , 'Mon'), sum(distance)
        FROM maintable
       GROUP BY id , to_char(times , 'Mon')
```
- Output pane:** Shows the results of the query in a table format.

	id character varying	to_char text	sum double precision
1	7079824-CALEND-703-Kupi>-04	Apr	78795.2098646022
2	7079824-CALEND-703-Kupi>-04	Mar	11152.1631362318
3	7079824-CALEND-703-Kupi>-04	May	80993.3605512418
4	7079825-CALEND-703-Kupi>-04	Apr	111635.273165609

Bottom Window:

- SQL Editor:** Contains the same SQL code as the top window, with the "EXPLAIN" keyword added before the first SELECT statement.

```
ALTER SYSTEM SET shared_buffers TO '1GB';
SELECT pg_reload_conf()
EXPLAIN( SELECT id, to_char(times , 'Mon'), sum(distance)
        FROM maintable
       GROUP BY id , to_char(times , 'Mon'))
```
- Output pane:** Shows the query plan in a table format.

	QUERY PLAN text
1	GroupAggregate (cost=1382322.21..1455720.99 rows=652434 width=59)
2	Group Key: id, (to char(times , 'Mon'::text))
3	-> Sort (cost=1382322.21..1398633.05 rows=6524336 width=59)
4	Sort Key: id, (to char(times , 'Mon'::text))
5	-> Seq Scan on maintable (cost=0.00..153250.20 rows=6524336 width=59)
- Status Bar:** Shows "11155 rows. 53.2 secs" at the bottom right.

iv.

The image displays two side-by-side DB2 Query windows, both titled "Query - db2 on postgres@localhost:5432".

Top Window (Left):

- SQL Editor:** Contains the following SQL code:

```
ALTER SYSTEM SET shared_buffers TO '1GB';
SELECT pg_reload_conf()

SELECT id, AVG(distance)
FROM maintable
GROUP BY id
```
- Output pane:** Shows the results of the query. The table has two columns: "id" (character varying) and "avg" (double precision).

	id	avg
	character varying	double precision
1	9056482-CALEND-040-Ιαθάπτο-07	594.062177024096
2	9230475-CALEND-891-Καθημερινή-07	302.82616009821

Bottom Window (Right):

- SQL Editor:** Contains the same SQL code as the top window, with an additional explain clause:

```
EXPLAIN (SELECT id, AVG(distance)
          FROM maintable
          GROUP BY id)
```
- Output pane:** Shows the query plan. The plan consists of three steps:
 - 1 HashAggregate (cost=169561.04..169580.84 rows=1584 width=51)
 - 2 Group Key: id
 - 3 -> Seq Scan on maintable (cost=0.00..136939.36 rows=6524336 width=51)
- Status Bar:** Displays "1596 rows. 2.1 secs".

V.

Query - db2 on postgres@localhost:5432 *

File Edit Query Favourites Macros View Help

SQL Editor Graphical Query Builder

Previous queries

```
ALTER SYSTEM SET shared_buffers TO '1GB';
SELECT pg_reload_conf();
SELECT region_id, AVG(distance)
FROM maintable
GROUP BY region_id
```

Output pane

Data Output Explain Messages History

	region_id	avg	smallint	double precision
1	34	563.112588281303		
2	43	558.214355422858		
3	32	571.0669701019967		
4	25	561.840616784074		

Scratch pad

Query - db2 on postgres@localhost:5432 *

File Edit Query Favourites Macros View Help

SQL Editor Graphical Query Builder

Previous queries

```
ALTER SYSTEM SET shared_buffers TO '1GB';
SELECT pg_reload_conf();
EXPLAIN(SELECT region_id, AVG(distance)
FROM maintable
GROUP BY region_id)
```

Output pane

Data Output Explain Messages History

QUERY PLAN	text
1	HashAggregate (cost=169561.04..169561.72 rows=54 width=10)
2	Group Key: region_id
3	-> Seq Scan on maintable (cost=0.00..136939.36 rows=6524336 width=10)

Scratch pad

54 rows. 1.9 secs

OK. DOS Ln 6, Col 10, Ch 150 3 rows. 15 msec

- Γ. Ερώτημα

Στο 3^ο ερώτημα της εργασίας χρειάστηκε, πριν πράξω οτιδήποτε, να εγκαταστήσω νεότερη έκδοση της PostgreSQL διότι δεν υπήρχε η εντολή max_parallel_workers_per_gather ή κάποια άλλη παρόμοια της στην έκδοση 9.5. Αυτός είναι και ο λόγος που στα screenshot έχει αλλάξει και το ui από pgadmin3 σε pgadmin4. Αφού έκανα την αλλαγή, εκτέλεσα την εντολή ALTER SYSTEM SET max_parallel_workers_per_gather = 30; και μετά έκανα επανεκκίνηση του server. Σε αυτό το ερώτημα παρατηρούνται αλλαγές διότι αυτή η εντολή αλλάζει την χρήση του επεξεργαστή που γίνεται από την PostgreSQL. Η μεγαλύτερη διαφορά παρατηρείται στο 3^ο ερώτημα διότι αυτό είχε τον μεγαλύτερο χρόνο εκτέλεσης. Στα Screenshot που θα δείτε παρακάτω έκρινα καλύτερο να φαίνεται ο χρόνος εκτέλεσης και όχι το αποτέλεσμα από εκείνα έχουν φανεί σε προηγούμενα screenshot. Αυτά είναι τα αποτελέσματα που είδα:

i.

```

SELECT A.id , SUM(A.distance) AS MaxDistance
FROM(
    SELECT id , times ,distance
    FROM maintable
    WHERE times < '2017-01-03 03:58:01'
        ) AS A
GROUP BY A.id
ORDER BY MaxDistance DESC
LIMIT 1;

```

Total query runtime: 1 secs.
1 rows retrieved.

```

explain(   SELECT A.id , SUM(A.distance) AS MaxDistance
  FROM(
    SELECT id , times ,distance
    FROM maintable
    WHERE times < '2017-01-03 03:58:01'
        ) AS A
GROUP BY A.id
ORDER BY MaxDistance DESC
LIMIT 1
)

```

EXPLAIN PLAN

Limit (cost=93229.20..93229.21 rows=1 width=51)
-> Sort (cost=93229.20..93233.16 rows=1584 width=51)
Sort Key: (sum(maintable.distance)) DESC
-> Finalize GroupAggregate (cost=93157.92..93221.28 rows=1584 width=51)
Group Key: maintable.id
-> Sort (cost=93157.92..93173.76 rows=6336 width=51)
Sort Key: maintable.id
-> Gather (cost=92108.39..92757.83 rows=6336 width=51)
Workers Planned: 4
-> Partial HashAggregate (cost=91108.39..91124.23 rows=1584 width=51)
Group Key: maintable.id
-> Parallel Seq Scan on maintable (cost=0.00..90002.51 rows=221176 width=51)
Filter: (times < '2017-01-03 03:58:01':timestamp without time zone)

ii.

The screenshot shows two pgAdmin 4 windows side-by-side, both connected to a PostgreSQL 9.5 server named DB2.

Top Window (Query Result):

- Browser:** Shows the server (DB2) and database (DB2) structure.
- Query Editor:** Contains the following SQL query:

```
1 SELECT AVG(distance) AS AverageDistance
2 FROM maintable
3 WHERE times >
4   (   select max(times) AS now
5     from maintable
6   ) -interval'1 month'
```
- Data Output:** Displays the result of the query:

averagedistance	double precision
566.045178187899	

Bottom Window (Query Plan):

- Browser:** Shows the server (DB2) and database (DB2) structure.
- Query Editor:** Contains the same SQL query as the top window.
- Explain Plan:** Displays the query plan in text format:

```
1 explain(  SELECT AVG(distance) AS AverageDistance
2   FROM maintable
3   WHERE times >
4     (   select max(times) AS now
5       from maintable
6     ) -interval'1 month'
7 )
```

QUERY PLAN

 - Aggregate (cost=255451.90..255451.91 rows=1 width=8)
 - InitPlan 1 (returns \$0)
 - > Finalize Aggregate (cost=91002.92..91002.93 rows=1 width=8)
 - > Gather (cost=91002.51..91002.91 rows=4 width=8)
 - Workers Planned: 4
 - > Partial Aggregate (cost=90002.51..90002.51 rows=1 width=8)
 - > Parallel Seq Scan on maintable maintable_1 (cost=0.00..86341.21 rows=1464520 width=8)
 - > Seq Scan on maintable (cost=0.00..159567.23 rows=1952694 width=8)
 - Filter: (times > (\$0 - '1 mon':interval))

iii.

The screenshot shows two pgAdmin 4 windows side-by-side, illustrating a query execution plan.

Top Window (PostgreSQL 9.5):

- Query:**

```
1 SELECT id, to_char(times , 'Mon'), sum(distance)
2 FROM maintable
3 GROUP BY id , to_char(times , 'Mon')
```
- Results:** Total query runtime: 24 secs.
11155 rows retrieved.

Bottom Window (PostgreSQL 9.5):

- Query:**

```
1 explain (SELECT id, to_char(times , 'Mon'), sum(distance)
2   FROM maintable
3   GROUP BY id , to_char(times , 'Mon'))
```
- EXPLAIN Plan:**

QUERY PLAN
text
Finalize GroupAggregate (cost=1109609.72..1140364.64 rows=585808 width=83)
Group Key: id, (to_char(times,'Mon')::text)
-> Sort (cost=1109609.72..1115467.80 rows=2343232 width=83)
Sort Key: id, (to_char(times,'Mon')::text)
-> Gather (cost=381144.97..637435.97 rows=2343232 width=83)
Workers Planned: 4
-> Partial GroupAggregate (cost=380144.97..402112.77 rows=585808 width=83)
Group Key: id, (to_char(times,'Mon')::text)
-> Sort (cost=380144.97..383806.27 rows=1464520 width=83)
Sort Key: id, (to_char(times,'Mon')::text)
-> Parallel Seq Scan on maintable (cost=0.00..90002.51 rows=1464520 width=83)

iv.

The screenshot shows two instances of pgAdmin 4 running side-by-side. Both instances are connected to a PostgreSQL 9.5 server named 'DB2'.

Top Window (Query Result):

- Browser:** Shows the database structure under 'DB2' including 'DB2', 'Cast', 'Cata', 'Ever', 'Exte', 'Fore', 'Lang', 'Sche', 'postgres', 'Login/Grou', and 'Tablespace'.
- Query Editor:** Displays the following SQL query:

```
1  SELECT id, AVG(distance)
2  FROM maintable
3  GROUP BY id;
```
- Messages Tab:** Shows the execution results:

Total query runtime: 1 secs.
1596 rows retrieved.

Bottom Window (Query Plan):

- Browser:** Same database structure as the top window.
- Query Editor:** Displays the same SQL query as the top window.
- Messages Tab:** Shows the execution results:

Total query runtime: 1 secs.
1596 rows retrieved.
- Explain Tab:** Shows the query plan in text format:

```
1  explain(  SELECT id, AVG(distance)
2    FROM maintable
3    GROUP BY id)|
```
- Results Tab:** Shows the expanded query plan with various stages and their costs and row counts.

V.

The screenshot shows two instances of the pgAdmin 4 interface. Both windows have the title bar "pgAdmin 4" and the menu bar "File", "Object", "Tools", "Help".

Top Window: This window is titled "DB2 on postgres@PostgreSQL 9.5". It contains a SQL editor pane with the following query:

```
1 SELECT region_id, AVG(distance)
  FROM maintable
 GROUP BY region_id
```

Below the SQL editor, the status bar indicates "Total query runtime: 1 secs." and "54 rows retrieved."

Bottom Window: This window is also titled "DB2 on postgres@PostgreSQL 9.5". It contains a SQL editor pane with the same query, followed by an "explain" command:

```
1 explain(SELECT region_id, AVG(distance)
  FROM maintable
 GROUP BY region_id
 )|
```

Below the SQL editor, the "Explain" tab is selected in the bottom navigation bar. The results show the query plan:

QUERY PLAN
text
Finalize GroupAggregate (cost=94694.32..94696.62 rows=54 width=10)
Group Key: region_id
-> Sort (cost=94694.32..94694.86 rows=216 width=34)
Sort Key: region_id
-> Gather (cost=94663.81..94685.95 rows=216 width=34)
Workers Planned: 4
-> Partial HashAggregate (cost=93663.81..93664.35 rows=54 width=34)
Group Key: region_id
-> Parallel Seq Scan on maintable (cost=0.00..86341.21 rows=1464520 width=10)

- Δ. Ερώτημα

Σε αυτό το ερώτημα κλήθηκα να δημιουργήσω κάποια ευρετήρια όπου αυτά χρειαζόντουσαν. Έτσι δημιούργησα ένα B δέντρο στην στήλη με τις ημερομηνίες διότι μόνο τα ερωτήματα I,II μπορούν να επωφεληθούν από την χρήση δέντρων αφού τα άλλα 3 ερωτήματα αναγκαστικά θα αναγκαστούν να διαβάσουν ολόκληρο τον πίνακα αφού δεν υπάρχει κάποια συνθήκη σε αυτά τα SELECT. Διάλεξα B δέντρο αφού χρειάζεται να μελετήσουμε διάστημα στην βάση και τα B δέντρα μας δίνουν αυτή την δυνατότητα. Η εντολή που χρησιμοποίησα ήταν αυτή: **CREATE INDEX ON mainTable (times DESC);** Και τα αποτελέσματα τα παρακάτω:

i.

```

pgAdmin 4
File Object Tools Help
Browser
Servers (2)
PostgreSQL 9.5
Databases DB2
DB2 on postgres@PostgreSQL 9.5
1 CREATE INDEX ON maintable (times DESC);
2
3     SELECT A.id , SUM(A.distance) AS MaxDistance
4     FROM(
5         SELECT id , times ,distance
6         FROM maintable
7         WHERE times < '2017-01-03 03:58:01'
8             ) AS A
9     GROUP BY A.id
10    ORDER BY MaxDistance DESC
11    LIMIT 1;
Data Output Explain Messages History
Total query runtime: 1 secs.
1 rows retrieved.

pgAdmin 4
File Object Tools Help
Browser
Servers (2)
PostgreSQL 9.5
Databases DB2
DB2 on postgres@PostgreSQL 9.5
1 CREATE INDEX ON maintable (times DESC);
2
3 explain(   SELECT A.id , SUM(A.distance) AS MaxDistance
4     FROM(
5         SELECT id , times ,distance
6         FROM maintable
7         WHERE times < '2017-01-03 03:58:01'
8             ) AS A
9     GROUP BY A.id |
10    ORDER BY MaxDistance DESC
11    LIMIT 1);
Data Output Explain Messages History
QUERY PLAN
text
Limit (cost=93229.18..93229.18 rows=1 width=51)
-> Sort (cost=93229.18..93233.14 rows=1584 width=51)
Sort Key: (sum(maintable.distance)) DESC
-> Finalize GroupAggregate (cost=93157.90..93221.26 rows=1584 width=51)
Group Key: maintable.id
-> Sort (cost=93157.90..93173.74 rows=6336 width=51)
Sort Key: maintable.id
-> Gather (cost=92108.36..92757.80 rows=6336 width=51)
Workers Planned: 4
-> Partial HashAggregate (cost=91108.36..91124.20 rows=1584 width=51)
Group Key: maintable.id
-> Parallel Seq Scan on maintable (cost=0.00..90002.48 rows=221176 width=51)
Filter: (times < '2017-01-03 03:58:01'::timestamp without time zone)

```

ii.

pgAdmin 4

File Object Tools Help

Browser

Servers (2) PostgreSQL 9.5 Databases (2) DB2

- Casts
- Catalog
- Event 1
- Extensi
- Foreignr
- Langua
- Schem
- postgres
- Login/Group R
- Tablespaces

DB2 on postgres@PostgreSQL 9.5

```

1 CREATE INDEX ON maintable (times DESC);
2
3 explain(   SELECT AVG(distance) AS AverageDistance
4           FROM maintable
5           WHERE times >
6               (   select max(times) AS now
7                   from maintable
8                   ) -interval'1 month')

```

Data Output Explain Messages History

Total query runtime: 1 secs.
1 rows retrieved.

pgAdmin 4

File Object Tools Help

Browser

Servers (2) PostgreSQL 9.5 Databases (2) DB2

- Casts
- Catalog
- Event 1
- Extensi
- Foreignr
- Langua
- Schem
- postgres
- Login/Group R
- Tablespaces

DB2 on postgres@PostgreSQL 9.5

```

1 CREATE INDEX ON maintable (times DESC);
2
3 explain(   SELECT AVG(distance) AS AverageDistance
4           FROM maintable
5           WHERE times >
6               (   select max(times) AS now
7                   from maintable
8                   ) -interval'1 month')

```

Data Output Explain Messages History

QUERY PLAN text

- Aggregate (cost=142422.41..142422.42 rows=1 width=8)
- InitPlan 2 (returns \$1)
- > Result (cost=0.51..0.52 rows=1 width=8)
- InitPlan 1 (returns \$0)
- > Limit (cost=0.43..0.51 rows=1 width=8)
- > Index Only Scan using maintable_times_idx on maintable maintable_1 (cost=0.43..453534.28 rows=5858074 width=8)
- Index Cond: (times IS NOT NULL)
- > Bitmap Heap Scan on maintable (cost=36553.79..137540.16 rows=1952691 width=8)
- Recheck Cond: (times > (\$1 - '1 mon':interval))
- > Bitmap Index Scan on maintable_times_idx (cost=0.00..36065.62 rows=1952691 width=0)
- Index Cond: (times > (\$1 - '1 mon':interval))

- **E. Ερώτημα**

Σε αυτό το ερώτημα έπρεπε να σπάσω το dataset σε κάποια partitions. Διάλεξα να το κάνω με range και να το χωρίσω ανά μήνα. Διάλεξα τον μήνα σαν κύρια στήλη για να κάνω αυτή την διαδικασία αφού και πάλι πιο πολύ θα επωφεληθούν τα δύο πρώτα queries από αυτό.

Αυτό που έκανα αρχικά είναι να φτιάξω ένα πίνακα ίδιο με αυτόν που ήδη είχα. Μετά έφτιαξα 8 υποπίνακες οι οποίοι κληρονομούσαν τον αρχικό αλλά ο καθένας είχε ένα συγκεκριμένο εύρος ημερομηνιών. Στην συνέχεια έφτιαξα ευρετήρια για κάθε ένα από τους υποπίνακες και ένα function και ένα trigger πριν το insert για να μπορέσω να βάζω τις σωστές εγγραφές στους αντίστοιχους πίνακες. Τέλος πέρασα τα δεδομένα στον πρώτο πίνακα που έφτιαξα και με την βοήθεια των trigger αυτά κατανεμήθηκαν στους αντίστοιχους υποπίνακες.

Ο κώδικας για όλα αυτά που προανέφερα είναι ο ακόλουθος:

```
CREATE TABLE ExersiseE (
    id varchar,
    times timestamp,
    distance double precision,
    region_id smallint
);
CREATE TABLE Ex1 (
    CHECK ( times >= '2016-12-01 00:00:00' AND times < '2017-01-01 00:00:00' )
) INHERITS (ExersiseE);

CREATE TABLE Ex2 (
    CHECK ( times >= '2017-01-01 00:00:00' AND times < '2017-02-01 00:00:00' )
) INHERITS (ExersiseE);

CREATE TABLE Ex3 (
    CHECK ( times >= '2017-02-01 00:00:00' AND times < '2017-03-01 00:00:00' )
) INHERITS (ExersiseE);

CREATE TABLE Ex4 (
    CHECK ( times >= '2017-03-01 00:00:00' AND times < '2017-04-01 00:00:00' )
) INHERITS (ExersiseE);

CREATE TABLE Ex5 (
    CHECK ( times >= '2017-04-01 00:00:00' AND times < '2017-05-01 00:00:00' )
) INHERITS (ExersiseE);

CREATE TABLE Ex6 (
    CHECK ( times >= '2017-05-01 00:00:00' AND times < '2017-06-01 00:00:00' )
) INHERITS (ExersiseE);

CREATE TABLE Ex7 (
    CHECK ( times >= '2017-06-01 00:00:00' AND times < '2017-07-01 00:00:00' )
) INHERITS (ExersiseE);
```

```

CREATE TABLE Ex8 (
    CHECK (times >= '2017-07-01 00:00:00' AND times < '2017-08-01 00:00:00' )
) INHERITS (ExersiseE);

CREATE INDEX Ex1_index ON Ex1 (times);
CREATE INDEX Ex2_index ON Ex2 (times);
CREATE INDEX Ex3_index ON Ex3 (times);
CREATE INDEX Ex4_index ON Ex4 (times);
CREATE INDEX Ex5_index ON Ex5 (times);
CREATE INDEX Ex6_index ON Ex6 (times);
CREATE INDEX Ex7_index ON Ex7 (times);
CREATE INDEX Ex8_index ON Ex8 (times);

CREATE OR REPLACE FUNCTION timesInsertTrigger()
RETURNS TRIGGER AS $$$
BEGIN
    IF (NEW.times >= '2016-12-01 00:00:00' AND NEW.times < '2017-01-01 00:00:00') THEN
        INSERT INTO Ex1 VALUES (NEW.*);
    ELSIF (NEW.times >= '2017-01-01 00:00:00' AND NEW.times < '2017-02-01 00:00:00') THEN
        INSERT INTO Ex2 VALUES (NEW.*);
    ELSIF (NEW.times >= '2017-02-01 00:00:00' AND NEW.times < '2017-03-01 00:00:00') THEN
        INSERT INTO Ex3 VALUES (NEW.*);
    ELSIF (NEW.times >= '2017-03-01 00:00:00' AND NEW.times < '2017-04-01 00:00:00') THEN
        INSERT INTO Ex4 VALUES (NEW.*);
    ELSIF (NEW.times >= '2017-04-01 00:00:00' AND NEW.times < '2017-05-01 00:00:00') THEN
        INSERT INTO Ex5 VALUES (NEW.*);
    ELSIF (NEW.times >= '2017-05-01 00:00:00' AND NEW.times < '2017-06-01 00:00:00') THEN
        INSERT INTO Ex6 VALUES (NEW.*);
    ELSIF (NEW.times >= '2017-06-01 00:00:00' AND NEW.times < '2017-07-01 00:00:00') THEN
        INSERT INTO Ex7 VALUES (NEW.*);
    ELSIF (NEW.times >= '2017-07-01 00:00:00' AND NEW.times < '2017-08-01 00:00:00') THEN
        INSERT INTO Ex8 VALUES (NEW.*);
    ELSE
        RAISE EXCEPTION 'Date out of range.';
    END IF;
    RETURN NULL;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER inserTimesTrigger
BEFORE INSERT ON ExersiseE
FOR EACH ROW EXECUTE PROCEDURE timesInsertTrigger();

COPY ExersiseE(id, times, distance, region_id)
FROM 'C:\Users\George\Desktop\unipi\5o\db2'
WITH CSV HEADER;

```

Τέλος, έτρεξα τα queries του ερωτήματος Α και είχα τα ακόλουθα αποτελέσματα(πάλι όπως και στα προηγούμενα ερωτήματα έκρινα σωστό να δείξω τους χρόνους και όχι το αποτέλεσμα καθεαυτό όπως και τα explain για κάθε query).

I.

The screenshot shows two instances of pgAdmin 4. The top instance is titled "DB2 on postgres@PostgreSQL 9.6" and contains the following SQL query:

```

1  SELECT A.id , SUM(A.distance) AS MaxDistance
2  FROM(
3      SELECT id , times ,distance
4      FROM ExersiseE
5      WHERE times < '2017-01-03 03:58:01'
6          ) AS A
7  GROUP BY A.id
8  ORDER BY MaxDistance DESC
9  LIMIT 1;

```

The bottom instance is also titled "DB2 on postgres@PostgreSQL 9.6" and contains the same query, preceded by an EXPLAIN keyword:

```

1  explain(   SELECT A.id , SUM(A.distance) AS MaxDistance
2    FROM(
3        SELECT id , times ,distance
4        FROM ExersiseE
5        WHERE times < '2017-01-03 03:58:01'
6            ) AS A
7    GROUP BY A.id
8    ORDER BY MaxDistance DESC
9    LIMIT 1)

```

The "Explain" window displays the following query plan:

QUERY PLAN
text
Limit (cost=29300.21..29300.22 rows=1 width=51)
-> Sort (cost=29300.21..29300.71 rows=200 width=51)
Sort Key: (sum(exersisee.distance)) DESC
-> Finalize GroupAggregate (cost=29292.71..29299.21 rows=200 width=51)
Group Key: exersisee.id
-> Sort (cost=29292.71..29294.21 rows=600 width=51)
Sort Key: exersisee.id
-> Gather (cost=29203.03..29265.03 rows=600 width=51)
Workers Planned: 3
-> Partial HashAggregate (cost=28203.03..28205.03 rows=200 width=51)
Group Key: exersisee.id
-> Append (cost=0.00..26796.93 rows=281219 width=51)
-> Parallel Seq Scan on exersisee (cost=0.00..0.00 rows=1 width=40)
Filter: (times < '2017-01-03 03:58:01'::timestamp without time zone)
-> Parallel Seq Scan on ex1 (cost=0.00..13424.54 rows=266203 width=51)
Filter: (times < '2017-01-03 03:58:01'::timestamp without time zone)
-> Parallel Seq Scan on ex2 (cost=0.00..13372.39 rows=15015 width=51)
Filter: (times < '2017-01-03 03:58:01'::timestamp without time zone)

- II. Το ερώτημα 2 λόγο της δομής του και του τρόπου που ορίσαμε τώρα τους πίνακες έχει ένα πολύ μεγάλο explain σε σχέση με τα υπόλοιπα, αυτό συμβαίνει διότι χρειάζεται να περάσει παραπάνω από μια φορά από την δομή που σχηματίσαμε παραπάνω. Το explain θα το δείξω σε τρία screenshot:

The screenshot shows the pgAdmin 4 interface. On the left, the Browser pane displays two servers: PostgreSQL 9.5 and PostgreSQL 9.6. Under PostgreSQL 9.6, the Databases section shows the DB2 database selected. The main window contains a SQL tab with the following query:

```
1  SELECT AVG(distance) AS AverageDistance
2  FROM ExersiseE
3  WHERE times >
4      (   select max(times) AS now
5          from ExersiseE
6          ) -interval'1 month'
```

Below the query, the Explain tab is active, showing the execution plan. A green box at the bottom right of the results area displays the execution statistics:

Total query runtime: 1 secs.
1 rows retrieved.

pgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

Browser

Servers (2)

- PostgreSQL 9.5
- PostgreSQL 9.6

Databases (1)

- DB2

Cast

Catalog

Ever

Extents

Foreign

Language

Schemas (1)

- postgres

Login/Groups

Tablespaces

Dashboard Properties SQL Statistics Dependencies Dependents Query- untitled *

DB2 on postgres@PostgreSQL 9.6

```
1 explain(SELECT AVG(distance) AS AverageDistance
2      FROM ExersiseE
3      WHERE times >
4          (    select max(times) AS now
5              from ExersiseE
```

Data Output Explain Messages History

QUERY PLAN text

- > Bitmap Index Scan on ex2_index (cost=0.00..6027.35 rows=273990 width=0)
- Index Cond: (times > (\$1 - '1 mon'::interval))
- > Bitmap Heap Scan on ex3 (cost=5540.75..18407.99 rows=248816 width=8)
- Recheck Cond: (times > (\$1 - '1 mon'::interval))
- > Bitmap Index Scan on ex3_index (cost=0.00..5478.55 rows=248816 width=0)
- Index Cond: (times > (\$1 - '1 mon'::interval))
- > Bitmap Heap Scan on ex4 (cost=6481.81..21798.61 rows=295920 width=8)
- Recheck Cond: (times > (\$1 - '1 mon'::interval))
- > Bitmap Index Scan on ex4_index (cost=0.00..6407.83 rows=295920 width=0)
- Index Cond: (times > (\$1 - '1 mon'::interval))
- > Bitmap Heap Scan on ex5 (cost=6184.76..20722.75 rows=281333 width=8)
- Recheck Cond: (times > (\$1 - '1 mon'::interval))
- > Bitmap Index Scan on ex5_index (cost=0.00..6114.43 rows=281333 width=0)
- Index Cond: (times > (\$1 - '1 mon'::interval))
- > Bitmap Heap Scan on ex6 (cost=6580.89..22131.60 rows=300447 width=8)
- Recheck Cond: (times > (\$1 - '1 mon'::interval))
- > Bitmap Index Scan on ex6_index (cost=0.00..6505.78 rows=300447 width=0)
- Index Cond: (times > (\$1 - '1 mon'::interval))
- > Bitmap Heap Scan on ex7 (cost=5970.88..19845.70 rows=268188 width=8)
- Recheck Cond: (times > (\$1 - '1 mon'::interval))
- > Bitmap Index Scan on ex7_index (cost=0.00..5903.84 rows=268188 width=0)
- Index Cond: (times > (\$1 - '1 mon'::interval))
- > Bitmap Heap Scan on ex8 (cost=197.43..650.24 rows=8921 width=8)
- Recheck Cond: (times > (\$1 - '1 mon'::interval))
- > Bitmap Index Scan on ex8_index (cost=0.00..195.20 rows=8921 width=0)
- Index Cond: (times > (\$1 - '1 mon'::interval))

pgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

Browser

Servers (2)

- PostgreSQL 9.5
- PostgreSQL 9.6

Databases (1)

- DB2

Cast

Catalog

Ever

External

Foreign

Language

Schemas (1)

- postgres

Login/Groups

Tablespaces

Dashboard Properties SQL Statistics Dependencies Dependents Query- untitled *

DB2 on postgres@PostgreSQL 9.6

```
1 explain(SELECT AVG(distance) AS AverageDistance
2      FROM ExersiseE
3      WHERE times >
4          (    select max(times) AS now
5              from ExersiseE
```

Data Output Explain Messages History

QUERY PLAN text

- > Seq Scan on exersisee exersisee_1 (cost=0.00..0.00 rows=1 width=8)
- Filter: (times IS NOT NULL)
- > Index Only Scan Backward using ex1_index on ex1 ex1_1 (cost=0.42..66724.50 rows=825230 width=8)
- Index Cond: (times IS NOT NULL)
- > Index Only Scan Backward using ex2_index on ex2 ex2_1 (cost=0.42..66514.30 rows=821969 width=8)
- Index Cond: (times IS NOT NULL)
- > Index Only Scan Backward using ex3_index on ex3 ex3_1 (cost=0.42..60432.83 rows=746448 width=8)
- Index Cond: (times IS NOT NULL)
- > Index Only Scan Backward using ex4_index on ex4 ex4_1 (cost=0.42..71587.55 rows=887760 width=8)
- Index Cond: (times IS NOT NULL)
- > Index Only Scan Backward using ex5_index on ex5 ex5_1 (cost=0.42..68049.87 rows=843999 width=8)
- Index Cond: (times IS NOT NULL)
- > Index Only Scan Backward using ex6_index on ex6 ex6_1 (cost=0.42..72693.50 rows=901341 width=8)
- Index Cond: (times IS NOT NULL)
- > Index Only Scan Backward using ex7_index on ex7 ex7_1 (cost=0.42..65158.34 rows=804564 width=8)
- Index Cond: (times IS NOT NULL)
- > Index Only Scan Backward using ex8_index on ex8 ex8_1 (cost=0.29..2119.92 rows=26763 width=8)
- Index Cond: (times IS NOT NULL)
- > Append (cost=0.00..144148.03 rows=1952693 width=8)
- > Seq Scan on exersisee (cost=0.00..0.00 rows=1 width=8)
- Filter: (times > (\$1 - '1 mon'::interval))
- > Bitmap Heap Scan on ex1 (cost=6104.27..20327.43 rows=275077 width=8)
- Recheck Cond: (times > (\$1 - '1 mon'::interval))
- > Bitmap Index Scan on ex1_index (cost=0.00..6035.51 rows=275077 width=0)
- Index Cond: (times > (\$1 - '1 mon'::interval))
- > Bitmap Heap Scan on ex2 (cost=6095.85..20263.70 rows=273990 width=8)
- Recheck Cond: (times > (\$1 - '1 mon'::interval))

pgAdmin 4

pgAdmin 4 File Object Tools Help

Browser Dashboard Properties SQL Statistics Dependencies Dependents Query- untitled *

Servers (2) PostgreSQL 9.5 PostgreSQL 9.6 Databases (DB2 Cast Cata Ever Exte Fore Lang Sch postres Login/Groups Tablespace

DB2 on postgres@PostgreSQL 9.6

```
1 explain(SELECT AVG(distance) AS AverageDistance
2   FROM ExersiseE
3   WHERE times >
4     (      select max(times) AS now
5       from ExersiseE
```

Data Output Explain Messages History

QUERY PLAN text

- > Seq Scan on exersisee exersisee_1 (cost=0.00..0.00 rows=1 width=8)
- Filter: (times IS NOT NULL)
- > Index Only Scan Backward using ex1_index on ex1 ex1_1 (cost=0.42..66724.50 rows=825230 width=8)
- Index Cond: (times IS NOT NULL)
- > Index Only Scan Backward using ex2_index on ex2 ex2_1 (cost=0.42..66514.30 rows=821969 width=8)
- Index Cond: (times IS NOT NULL)
- > Index Only Scan Backward using ex3_index on ex3 ex3_1 (cost=0.42..60432.83 rows=746448 width=8)
- Index Cond: (times IS NOT NULL)
- > Index Only Scan Backward using ex4_index on ex4 ex4_1 (cost=0.42..71587.55 rows=887760 width=8)
- Index Cond: (times IS NOT NULL)
- > Index Only Scan Backward using ex5_index on ex5 ex5_1 (cost=0.42..68049.87 rows=843999 width=8)
- Index Cond: (times IS NOT NULL)
- > Index Only Scan Backward using ex6_index on ex6 ex6_1 (cost=0.42..72693.50 rows=901341 width=8)
- Index Cond: (times IS NOT NULL)
- > Index Only Scan Backward using ex7_index on ex7 ex7_1 (cost=0.42..65158.34 rows=804564 width=8)
- Index Cond: (times IS NOT NULL)
- > Index Only Scan Backward using ex8_index on ex8 ex8_1 (cost=0.29..2119.92 rows=26763 width=8)
- Index Cond: (times IS NOT NULL)
- > Append (cost=0.00..144148.03 rows=1952693 width=8)
- > Seq Scan on exersisee (cost=0.00..0.00 rows=1 width=8)
- Filter: (times > (\$1 - '1 mon'::interval))
- > Bitmap Heap Scan on ex1 (cost=6104.27..20327.43 rows=275077 width=8)
- Recheck Cond: (times > (\$1 - '1 mon'::interval))
- > Bitmap Index Scan on ex1_index (cost=0.00..6035.51 rows=275077 width=0)
- Index Cond: (times > (\$1 - '1 mon'::interval))
- > Bitmap Heap Scan on ex2 (cost=6095.85..20263.70 rows=273990 width=8)
- Recheck Cond: (times > (\$1 - '1 mon'::interval))

III.

```

pgAdmin 4
File Object Tools Help
Browser
Servers (2)
PostgreSQL 9.5
PostgreSQL 9.6
Databases (2)
DB2
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Schemas (1)
public
Collations
Domains
1
2
3
4
SELECT id, to_char(times , 'Mon'), sum(distance)
FROM exersisee
GROUP BY id , to_char(times , 'Mon')
LIMIT 1;

Data Output Explain Messages History
Total query runtime: 12 secs.
1 rows retrieved.

```



```

pgAdmin 4
File Object Tools Help
Browser
Servers (2)
PostgreSQL 9.5
PostgreSQL 9.6
Databases (2)
DB2
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Schemas (1)
public
Collations
Domains
1
2
3
4
explain(SELECT id, to_char(times , 'Mon'), sum(distance)
        FROM exersisee
        GROUP BY id , to_char(times , 'Mon')
        LIMIT 1)

Data Output Explain Messages History
QUERY PLAN
text
Limit (cost=558885.56..558885.61 rows=1 width=83)
-> Finalize GroupAggregate (cost=558885.56..560585.56 rows=40000 width=83)
Group Key: exersisee.id, (to_char(exersisee.times, 'Mon')::text)
-> Sort (cost=558885.56..559185.56 rows=120000 width=83)
Sort Key: exersisee.id, (to_char(exersisee.times, 'Mon')::text)
-> Gather (cost=510848.04..543018.46 rows=120000 width=83)
Workers Planned: 3
-> Partial GroupAggregate (cost=509848.04..530018.46 rows=40000 width=83)
Group Key: exersisee.id, (to_char(exersisee.times, 'Mon')::text)
-> Sort (cost=509848.04..514765.64 rows=1967042 width=83)
Sort Key: exersisee.id, (to_char(exersisee.times, 'Mon')::text)
-> Result (cost=0.00..115959.44 rows=1967042 width=83)
-> Append (cost=0.00..91371.41 rows=1967042 width=59)
-> Parallel Seq Scan on exersisee (cost=0.00..0.00 rows=1 width=48)
-> Parallel Seq Scan on ex1 (cost=0.00..12759.03 rows=266203 width=59)
-> Parallel Seq Scan on ex2 (cost=0.00..12709.51 rows=265151 width=59)
-> Parallel Seq Scan on ex3 (cost=0.00..12245.20 rows=311020 width=59)
-> Parallel Seq Scan on ex4 (cost=0.00..13741.74 rows=286374 width=59)
-> Parallel Seq Scan on ex5 (cost=0.00..13040.58 rows=272258 width=59)
-> Parallel Seq Scan on ex6 (cost=0.00..13951.55 rows=290755 width=59)
-> Parallel Seq Scan on ex7 (cost=0.00..12447.37 rows=259537 width=59)
-> Parallel Seq Scan on ex8 (cost=0.00..476.43 rows=15743 width=55)

```

IV.

The screenshot shows two instances of pgAdmin 4 running side-by-side. Both instances are connected to a PostgreSQL 9.6 database named 'DB2'.

Top Window (Query Result):

- Servers:** PostgreSQL 9.5, PostgreSQL 9.6
- Databases:** DB2, Cast, Cata, Ever, Exte, Fore, Lang, Sch, postgres
- Query:**

```
1  SELECT id, AVG(distance)
2  FROM ExersiseE
3  GROUP BY id
```
- Messages:** Total query runtime: 1 secs.
1596 rows retrieved.

Bottom Window (Query Plan):

- Servers:** PostgreSQL 9.5, PostgreSQL 9.6
- Databases:** DB2, Cast, Cata, Ever, Exte, Fore, Lang, Sch, postgres
- Query:**

```
1  explain(  SELECT id, AVG(distance)
2    FROM ExersiseE
3    GROUP BY id)
```
- Messages:** Total query runtime: 1 secs.
1596 rows retrieved.
- QUERY PLAN:**

text
Finalize GroupAggregate (cost=102296.31..102303.31 rows=200 width=51)
Group Key: exersisee.id
-> Sort (cost=102296.31..102297.81 rows=600 width=75)
Sort Key: exersisee.id
-> Gather (cost=102206.62..102268.62 rows=600 width=75)
Workers Planned: 3
-> Partial HashAggregate (cost=101206.62..101208.62 rows=200 width=75)
Group Key: exersisee.id
-> Append (cost=0.00..91371.41 rows=1967042 width=51)
-> Parallel Seq Scan on exersisee (cost=0.00..0.00 rows=1 width=40)
-> Parallel Seq Scan on ex1 (cost=0.00..12759.03 rows=266203 width=51)
-> Parallel Seq Scan on ex2 (cost=0.00..12709.51 rows=265151 width=51)
-> Parallel Seq Scan on ex3 (cost=0.00..12245.20 rows=311020 width=51)
-> Parallel Seq Scan on ex4 (cost=0.00..13741.74 rows=286374 width=51)
-> Parallel Seq Scan on ex5 (cost=0.00..13040.58 rows=272258 width=51)
-> Parallel Seq Scan on ex6 (cost=0.00..13951.55 rows=290755 width=51)
-> Parallel Seq Scan on ex7 (cost=0.00..12447.37 rows=259537 width=51)
-> Parallel Seq Scan on ex8 (cost=0.00..476.43 rows=15743 width=47)

V.

The screenshot shows two instances of pgAdmin 4 running side-by-side. Both instances are connected to a PostgreSQL 9.6 database named 'DB2'.

Top Window (Query Result):

- Browser:** Shows the database structure with 'Databases' expanded, revealing 'DB2' and 'postgres'.
- Query Editor:** Contains the following SQL query:

```
1 SELECT region_id, AVG(distance)
2 FROM ExersiseE
3 GROUP BY region_id
```
- Messages Tab:** Displays the execution results:

Total query runtime: 1 secs.
54 rows retrieved.

Bottom Window (Query Plan):

- Browser:** Shows the same database structure as the top window.
- Query Editor:** Contains the same SQL query as the top window.
- Data Output Tab:** Shows the query plan in text format:

```
explain( SELECT region_id, AVG(distance)
  FROM ExersiseE
  GROUP BY region_id )
```
- Explain Tab:** Provides a detailed breakdown of the query execution plan, listing various stages like Finalize GroupAggregate, Sort, Gather, and Parallel Seq Scan operations across multiple workers.
- Messages Tab:** Displays the execution results:

Total query runtime: 1 secs.
54 rows retrieved.

Mépoç 2º

- A. Στο πρώτο ερώτημα δημιουργησα μερικά στατιστικά για την βάση, συγκεκριμένα το πρώτο μετρά το πλήθος των μοναδικών αυτοκινήτων, το δεύτερο μετρά το πλήθος των μοναδικών περιοχών και το τρίτο μετρά την συνολική απόσταση ανά περιοχή. Για το κάθε ένα έπραξα παρόμοια οπότε θα περιγράψω την λειτουργεί μόνο του πρώτου:

Αρχικά δημιουργησα ένα configuration, έθεσα ένα master και επίσης δημιουργησα το sparkcontext. Στην συνέχεια πέρασα το αρχείο με την βοήθεια του context στο JavaRDD και του αγνόησα την πρώτη γραμμή δηλαδή το header. Μετά έφτιαξα ένα PairRDD για να μπορέσω να κάνω το mapping. Για το mapping χρησιμοποιήσα την .mapToPair και δημιουργησα ένα Tuple2 το οποίο γέμισα με την πρώτη και την τρίτη εγγραφή κάθε γραμμής αντίστοιχα. Τέλος για το reduceByKey έλαβα σαν key το id που είχα περάσει σαν πρώτο στοιχείο προηγουμένως και έκανα reduce με αθροιστική συνθήκη, ουσιαστικά η σύνθεση δεν έπαιζε πολύ ρόλο αφού στο τέλος θα έκανα count τις συνολικές γραμμές για να μου πει πόσοι είναι οι χρήστες.

Παραθέτω Screenshot στην συνέχεια:

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer (Package Expl):** Shows projects like DB2, question1, and SparkLabs.
- Editor:** Displays the Java code for `statistic1.java`. The code reads a CSV file, splits each row into a tuple, and then reduces it to count unique vehicles.
- Console:** Shows the terminal output of the application's execution. It includes log messages from the TaskScheduler and DAGScheduler, and informational messages from SparkUI, MemoryStore, BlockManager, and OutputCommitCoordinator.

```
<terminated>statistic1 [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Jan 31, 2018, 1:41:47 AM)
18/01/31 01:42:16 INFO TaskSchedulerImpl: Removed TaskSet 2.0, whose tasks have all completed, from pool
Number of unique vehicles: 1596
18/01/31 01:42:16 INFO DAGScheduler: ResultStage 2 (count at statistic1.java:27) finished in 0.481 s
18/01/31 01:42:16 INFO DAGScheduler: Job 1 finished: count at statistic1.java:27, took 14.247558 s
18/01/31 01:42:16 INFO SparkUI: Stopped Spark web UI at http://192.168.1.2:4040
18/01/31 01:42:16 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/01/31 01:42:16 INFO MemoryStore: MemoryStore cleared
18/01/31 01:42:16 INFO BlockManager: BlockManager stopped
18/01/31 01:42:17 INFO BlockManagerMaster: BlockManagerMaster stopped
18/01/31 01:42:17 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/01/31 01:42:17 INFO SparkContext: Successfully stopped SparkContext
18/01/31 01:42:17 INFO ShutdownHookManager: Shutdown hook called
18/01/31 01:42:17 INFO ShutdownHookManager: Deleting directory /tmp/spark-0e455fda-c1dc-42ab-ae8c35bb6e13
```

eclipse-workspace - DB2/src/statistic2.java - Eclipse

```

import org.apache.spark.SparkConf;
public class statistic2 {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        SparkConf conf = new SparkConf();
        conf.setAppName("BigData example").setMaster("local[*]");
        JavaSparkContext jsc = new JavaSparkContext(conf);

        JavaRDD<String> lines = jsc.textFile(System.getProperty("user.dir")+"/src/db2_project_data.csv");

        String header = lines.first();
        lines = lines.filter(row -> !row.equalsIgnoreCase(header));

        JavaPairRDD<String, Integer> consumption = lines.mapToPair(s -> {
            String[] foo= s.split(",");
            return new Tuple2<>(foo[3], 1);
        });

        JavaPairRDD<String, Integer> result = consumption.reduceByKey((x, y) -> x+y);
        System.out.println("Number of unique regions: " + result.count());
        jsc.close();
    }
}

```

Problems Javadoc Declaration Console

```

terminated> statistic2 [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Jan 31, 2018, 1:44:00 AM)
18/01/31 01:44:07 INFO TaskSetManager: Finished task 13.0 in stage 2.0 (TID 29) in 22 ms on localhost (executor driver) (15/15)
Number of unique regions: 54
18/01/31 01:44:07 INFO TaskSchedulerImpl: Removed TaskSet 2.0, whose tasks have all completed, from pool
18/01/31 01:44:07 INFO DAGScheduler: ResultStage 2 (count at statistic2.java:27) finished in 0.145 s
18/01/31 01:44:07 INFO DAGScheduler: Job 1 finished: count at statistic2.java:27, took 3.210561 s
18/01/31 01:44:07 INFO SparkUI: Stopped Spark web UI at http://192.168.1.2:4040
18/01/31 01:44:07 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/01/31 01:44:07 INFO MemoryStore: MemoryStore cleared
18/01/31 01:44:07 INFO BlockManager: BlockManager stopped
18/01/31 01:44:07 INFO BlockManagerMaster: BlockManager stopped
18/01/31 01:44:07 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/01/31 01:44:07 INFO SparkContext: Successfully stopped SparkContext
18/01/31 01:44:07 INFO ShutdownHookManager: Shutdown hook called
18/01/31 01:44:07 INFO ShutdownHookManager: Deleting directory /tmp/spark-7ffce734-eed9-48b9-aa78-1f2926c1794f

```

Writable Smart Insert 15:104

eclipse-workspace - DB2/src/statistic3.java - Eclipse

```

import org.apache.spark.SparkConf;
public class statistic3 {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        SparkConf conf = new SparkConf();
        conf.setAppName("BigData example").setMaster("local[*]");
        JavaSparkContext jsc = new JavaSparkContext(conf);

        JavaRDD<String> lines = jsc.textFile(System.getProperty("user.dir")+"/src/db2_project_data.csv");

        String header = lines.first();
        lines = lines.filter(row -> !row.equalsIgnoreCase(header));

        JavaPairRDD<String, Double> consumption = lines.mapToPair(s -> {
            String[] foo= s.split(",");
            return new Tuple2<>(foo[3],Double.parseDouble(foo[2]));
        });

        JavaPairRDD<String, Double> result = consumption.reduceByKey((x, y) -> x+y);
        result.foreach(x->System.out.println("RegionId:\t"+x._1+"\tTotal distance per region:\t"+x._2));
    }
}

```

Problems Javadoc Declaration Console

```

terminated> statistic3 [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Jan 31, 2018, 1:45:35 AM)
RegionId: 14 Total distance per region: 6.117605143661157E7
RegionId: 32 Total distance per region: 6.216349503132052E7
RegionId: 41 Total distance per region: 6.044223378968343E7
RegionId: 50 Total distance per region: 5.995520220213528E7
RegionId: 23 Total distance per region: 6.068839291674198E7
18/01/31 01:45:42 INFO TaskSetManager: Finished task 14.0 in stage 2.0 (TID 30) in 21 ms on localhost (executor driver) (14/15)
18/01/31 01:45:42 INFO Executor: Finished task 11.0 in stage 2.0 (TID 27). 1095 bytes result sent to driver
18/01/31 01:45:42 INFO TaskSetManager: Finished task 11.0 in stage 2.0 (TID 27) in 31 ms on localhost (executor driver) (15/15)
18/01/31 01:45:42 INFO DAGScheduler: Removed TaskSet 2.0, whose tasks have all completed, from pool
18/01/31 01:45:42 INFO DAGScheduler: ResultStage 2 (foreach at statistic3.java:25) finished in 0.145 s
18/01/31 01:45:42 INFO DAGScheduler: Job 1 finished: foreach at statistic3.java:25, took 3.792846 s
18/01/31 01:45:42 INFO SparkUI: Stopped Spark web UI at http://192.168.1.2:4040
18/01/31 01:45:42 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/01/31 01:45:42 INFO MemoryStore: MemoryStore cleared
18/01/31 01:45:42 INFO BlockManager: BlockManager stopped
18/01/31 01:45:42 INFO BlockManagerMaster: BlockManagerMaster stopped

```

Writable Smart Insert 13:104

B. Σε αυτό το ερώτημα πήρα σαν βάση το προηγούμενο ερώτημα και το μεταποίησα όπου αυτό χρειάστηκε. Πιο συγκεκριμένα, τα αρχικά βήματα μέχρι το map είναι τα ίδια. Στο map αφού δεν έχω τύπο με τρία στοιχεία έπρεπε να φτιάξω ένα μόνος μου. Οπότε, έβαλα το απλό PairRDD<string, > και στην δεύτερη θέση χρησιμοποίησα το Tuple2 για να μπορέσω να φτιάξω δυο στοιχεία από εκεί που είχα ένα. Έτσι με το mapToPair έβαλα σε κάθε ένα την πρώτη και την τρίτη τιμή κάθε γραμμής και στην τρίτη θέση που δημιούργησα έβαλα έναν άσσο για να μπορέσω μετά να βρω το count των ίδιων γραμμών ώστε να διαιρέσω με αυτό το sum και να βγει ο M.O.. Τέλος μετά το reduce εμφάνισα τα id και το πηλίκο της προαναφερθείσας διαιρεσης. Παραθέτω Screenshot στην συνέχεια:

eclipse-workspace - DB2/src/question2.java - Eclipse

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer (Package Expl):** Shows the project structure with files like question2.java, statistic1.java, statistic2.java, statistic3.java, TransformationsExample.java, and question3.java.
- Code Editor:** Displays the Java code for question2.java, which reads a CSV file, processes it using mapToPair and reduceByKey, and prints the results to the console.
- Console Output:** Shows the execution results of the application, including the calculated average values for each car ID and some INFO log messages from the executor and task manager.

```
import org.apache.spark.SparkConf;
public class question2 {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf();
        conf.setAppName("BigData example").setMaster("local[*]");
        JavaSparkContext jsc = new JavaSparkContext(conf);

        JavaRDD<String> lines = jsc.textFile(System.getProperty("user.dir")+"src/db2_project_data.csv");
        String header = lines.first();
        lines = lines.filter(row -> !row.equalsIgnoreCase(header));

        JavaPairRDD<String, Tuple2<Double,Double>> consumption = lines.mapToPair(s -> {
            String[] foo= s.split(",");
            return new Tuple2<Double,Double>(Double.parseDouble(foo[2]),1.0);
        });

        JavaPairRDD<String, Tuple2<Double,Double>> result = consumption.reduceByKey((x, y) -> {
            return new Tuple2<Double,Double>(x._1+y._1, x._2+y._2);
        });

        result.foreach(x->System.out.println("CarId:\t"+x._1 +"\t\tAVG: "+ x._2./x._2));
    }
}
```

CarId	Avg
9257036-CALEND-040-Καθημερινή-11	592.730432869218
9759668-CALEND-304-305-Σαρβάτο-10	668.9619495467546
9586378-CALEND-217-229-Σαρβάτο-14	577.8611835070528
8135852-CALEND-049-Σάρβατο-06	505.31166128549927
9056395-CALEND-040-Σάρβατο-07	576.6752330169395
8119255-CALEND-040-Κυριακή-07	572.92303432239802
9787506-CALEND-217-229-Καθημερινή-16	507.66443582230573
9759672-CALEND-304-305-Σαρβάτο-10	668.2001259280814

```
18/01/31 01:43:31 INFO Executor: Finished task 14.0 in stage 2.0 (TID 30). 1095 bytes result sent to driver
18/01/31 01:43:31 INFO TaskSetManager: Finished task 14.0 in stage 2.0 (TID 30) in 34 ms on localhost (executor driver) (13/15)
18/01/31 01:43:31 INFO TaskSetManager: Finished task 12.0 in stage 2.0 (TID 28) in 45 ms on localhost (executor driver) (14/15)
18/01/31 01:43:31 INFO Executor: Finished task 13.0 in stage 2.0 (TID 29). 1095 bytes result sent to driver
18/01/31 01:43:31 INFO TaskSetManager: Finished task 13.0 in stage 2.0 (TID 29) in 49 ms on localhost (executor driver) (15/15)
18/01/31 01:43:31 INFO TaskSchedulerImpl: Removed TaskSet 2.0, whose tasks have all completed, from pool
18/01/31 01:43:31 INFO DAGScheduler: ResultStage 2 (foreach at question2.java:27) finished in 0.250 s
18/01/31 01:43:31 INFO DAGScheduler: Job 1 finished: foreach at question2.java:27, took 4.487589 s
```

C. Σε αυτό το ερώτημα δοκίμασα δυο διαφορετικές προσεγγίσεις. Για αρχή χρησιμοποίησα την εντολή .repartition() και παρατήρησα ότι όσο μεγάλωνα την μεταβλητή της παρένθεσης τόσο ανέβαιναν οι χρόνοι εκτέλεσης. Έτσι στην συνέχεια χρησιμοποίησα το partitionBy(new HashPartitioner(4)) με το οποίο δεν παρατήρησα αισθητές διαφορές από το αρχικό πλάνο πέρα από κάτι μικρές αυξήσεις πάλι του χρόνου. Στο εκτελέσιμο αρχείο υπάρχουν και οι δύο υλοποιήσεις αλλά η μια είναι σε σχόλιο. Παραθέτω Screenshot στην συνέχεια:

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - DB2/src/question3.java - Eclipse
- Toolbar:** Standard Eclipse toolbar.
- Left Sidebar:** Package Explorer showing projects DB2, question1, and SparkLabs.
- Central Area:**
 - Code Editor:** Displays the Java code for `question3.java`. The code reads a CSV file, filters the header, repartitions the data, and calculates the average of two columns using a reduceByKey operation.
 - Console Tab:** Shows the execution output of the application.
- Bottom Status Bar:** Shows the status "Writable" and the current time "28:17".

```

eclipse-workspace - DB2/src/question3.java - Eclipse
-----
question2.java  statistic1.java  statistic2.java  statistic3.java  TransformationsExample.java  question3.java
-----
14 import org.apache.spark.HashPartitioner;[]
15 public class question3 {
16     public static void main(String[] args) {
17         SparkConf conf = new SparkConf();
18         conf.setAppName("BigData example").setMaster("local[*]");
19         JavaSparkContext jsc = new JavaSparkContext(conf);
20
21         JavaRDD<String> lines = jsc.textFile(System.getProperty("user.dir")+"/src/db2_project_data.csv");
22
23         String header = lines.first();
24         lines = lines.filter(row -> !row.equalsIgnoreCase(header));
25         lines = lines.repartition(4);
26
27         JavaPairRDD<String, Tuple2<Double,Double>> consumption = lines.mapToPair(s -> {
28             String[] foo= s.split(",");
29             return new Tuple2<>(foo[0], new Tuple2<Double,Double>(Double.parseDouble(foo[2]),1.0));
30         });
31         JavaPairRDD<String, Tuple2<Double,Double>> result = consumption.reduceByKey((x, y) -> {
32             return new Tuple2<Double,Double>(x._1+y._1, x._2+y._2);
33         });
34         result.foreach(x->System.out.println("CarId:\t"+x._1 +"\tAVG: "+ x._2/x._2));
35
36         jsc.close();
37     }
38 }

```

Console Output:

```

<terminated> question3 [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Jan 31, 2018, 1:47:08 AM)
18/01/31 01:47:20 INFO TaskSchedulerImpl: Removed TaskSet 3.0, whose tasks have all completed, from pool
18/01/31 01:47:20 INFO DAGScheduler: ResultStage 3 (foreach at question3.java:29) finished in 0.133 s
18/01/31 01:47:20 INFO DAGScheduler: Job 1 finished: foreach at question3.java:29, took 8.600772 s
18/01/31 01:47:20 INFO SparkUI: Stopped Spark web UI at http://192.168.1.2:4040
18/01/31 01:47:21 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/01/31 01:47:21 INFO MemoryStore: MemoryStore cleared
18/01/31 01:47:21 INFO BlockManager: BlockManager stopped
18/01/31 01:47:21 INFO BlockManagerMaster: BlockManagerMaster stopped
18/01/31 01:47:21 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/01/31 01:47:21 INFO SparkContext: Successfully stopped SparkContext
18/01/31 01:47:21 INFO ShutdownHookManager: Shutdown hook called
18/01/31 01:47:21 INFO ShutdownHookManager: Deleting directory /tmp/spark-859ffb4a-11bd-4672-9686-f6a5ce6eb0ed

```

eclipse-workspace - DB2/src/question3.java - Eclipse

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer (Package Expl.)**: Shows packages DB2, question1, and SparkLabs.
- Editor Area**: Displays the Java code for `question3.java`. The code reads a CSV file, filters rows, and calculates the average of a specific column.
- Console Tab**: Shows the execution output of the application. It lists car IDs and their corresponding average values, followed by Spark executor logs and DAGScheduler information.

```
10
11     SparkConf conf = new SparkConf();
12     conf.setAppName("BigData example").setMaster("local[*]");
13     JavaSparkContext jsc = new JavaSparkContext(conf);
14
15     JavaRDD<String> lines = jsc.textFile(System.getProperty("user.dir")+"/"+src/db2_project_data.csv");
16
17     String header = lines.first();
18     lines = lines.filter(row -> !row.equalsIgnoreCase(header));
19     //lines = lines.repartition(4);
20
21     JavaPairRDD<String, Tuple2<Double,Double>> consumption = lines.mapToPair(s -> {
22         String[] foo= s.split(",");
23         return new Tuple2<Double,Double>(Double.parseDouble(foo[2]),1.0));
24     });
25     JavaPairRDD<String, Tuple2<Double,Double>> result = consumption.reduceByKey(x, y) -> {
26         return new Tuple2<Double,Double>(x._1+y._1, x._2+y._2);
27     }).partitionBy(new HashPartitioner(4));
28
29     result.foreach(x->System.out.println("CarId:\t"+x._1 +"\tAVG: "+ x._2/x._2));
30
    inc.close();
```

CarId	Avg
7995343-CALEND-304-305-Κυριακή-07	660.7716280062003
9740869-CALEND-049-Κυριακή-12	523.4745387380298
9759630-CALEND-304-305-Σάββατο-10	659.066801386731
9741626-CALEND-304-305-Καθημερινή-11	654.6392126631696
9741644-CALEND-304-305-Καθημερινή-11	667.7607291962138
9741608-CALEND-304-305-Καθημερινή-11	656.8782828195792
9761257-CALEND-829-Καθημερινή-10	263.8972430939403
7995390-CALEND-304-305-Κυριακή-07	696.1845085062015
8119182-CALEND-040-Κυριακή-07	611.2734453696326

18/01/31 01:49:00 INFO Executor: Finished task 0.0 in stage 3.0 (TID 31). 966 bytes result sent to driver
18/01/31 01:49:00 INFO TaskSetManager: Finished task 0.0 in stage 3.0 (TID 31) in 71 ms on localhost (executor driver) (4/4)
18/01/31 01:49:00 INFO TaskSchedulerImpl: Removed TaskSet 3.0, whose tasks have all completed, from pool
18/01/31 01:49:00 INFO DAGScheduler: ResultStage 3 (foreach at question3.java:29) finished in 0.069 s
18/01/31 01:49:00 INFO DAGScheduler: Job 1 finished: foreach at question3.java:29, took 4.758228 s
18/01/31 01:49:00 INFO SparkUI: Stopped Spark web UI at http://192.168.1.2:4040
18/01/31 01:49:00 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/01/31 01:49:00 INFO MemoryStore: MemoryStore cleared